MICROPROCESSOR BASED NON-LINEAR ADAPTIVE CONTROLLER

by

K. L. YUNG

BSc, MSc, DIC, CEng, MIEE

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy of the Council for National Academic Awards.

This work was carried out in the School of Electrical and Electronic Engineering, Plymouth Polytechnic.

February 1985

TO

MY WIFE

AND

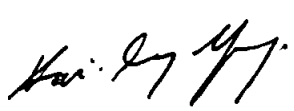
MY PARENTS

ACKNOWLEDGEMENTS

I would like to thank Professor Eamonn McQuade and Mr. Allan Roberts for their supervision and guidance during the course of this research and the preparation of this thesis. I am indebted to technicians in Plymouth Polytechnic who had help me in building some of the hardware. Particular thanks are due to my wife Wai Yee for her understanding during the preparation of this thesis and her typing of this thesis. Last but not the least, I would like to thank John Heathcoat & Co. Ltd. for the provision of the nylon crimping plant for research.

DECLARATION

No part of this thesis has been submitted in support of an application for another award of the CNAA or qualifications at any university or other institution of learning.



Kai Leung YUNG.

A B S T R A C T

MICROPROCESSOR BASED NON-LINEAR ADAPTIVE CONTROLLER
by K. L. YUNG

The advent of microprocessors has created the possibility of developing low cost adaptive controllers for small process plants which in the past badly needed but could not afford such controllers. To examine the practicality of developing advanced low cost microprocessor based controller, this thesis describes the development of a non-linear adaptive controller for a nylon crimping plant which is a typical example of small process plants.

In order to test the algorithm on site, an algorithm development/implement device basing on a novel multi-tasking concept was developed. This novel microprocessor based device can perform program development, on-line algorithm test and data logging at the same time, while, still maintaining its small size for easy transportation. When the control algorithm was fully developed and tested, a low cost dedicated controller using an Intel 8085 processor was designed to house the algorithm and as a direct replacement of the original analogue controller.

CONTENTS

Contents

Contents

1. INTRODUCTION

This thesis explains aspects of the application of microprocessors in process control. It considers the potential for implementing adaptive controllers, describes the development of a novel flexible algorithm evaluation system and investigates the control of a novel fibre crimping plant.

Recent publicity has thrown light on applications of microprocessors in replacing human operators, creating fears of mass unemployment. However, apart from this issue the more important role of microprocessors is in improving the performance of processes which are difficult or impossible to be achieved by manual operations. With the continual increase in computing power and the reduction in cost, it is possible to implement more intelligent process controllers.

In the past most process control operations were performed by analogue controllers. Due to the high cost in implementing more sophisticated algorithms, most analogue controllers are limited to the very basic 3-term control functions. With minicomputers coming onto the market, analogue controllers were gradually replaced by minicomputers in some industries. Normally, one minicomputer would replace several analogue controllers by allocating computing time slots for each control loop. Although this has definite advantages in improving the start up and shut down of processes and in being able to perform more advanced control, its disadvantages are cabling cost (1) and

complete plant shut down due to computer failure. Obviously, using a second computer as stand-by would reduce the probability of complete plant shut down due to computer failure. However, this would double the cost. With the advent of micro-processors, a lot of processes have now moved from hierarchical control to distributed control (2). Distributed control gives all the advantages of Direct Digital Control while localising control calculations down to single loop level. The beauty of distributed control is that the whole system is built up of individual blocks such that the total size of the system is not governed by a central unit. Multivariable control is still possible by providing communication links between each controller.

Until very recently most of the controllers on the market uses the very basic 3-term control functions. Even for micro-processor based controllers, the discrete 3-term controller is still widely used. This points to the fact that 3-term control functions are easier for the process operator to understand, and, quite rightly that most processes can be controlled by a 3-term controller satisfactorily although not optimally. Furthermore, with the limited computing power of microprocessors, more complex control algorithm may not be feasible because of the hard real time calculation requirement.

In the meantime, due to the oil crisis and the world shortage of energy and raw material, the price of raw material had highlighted the requirement for industrial processes to be operated close to optimal. This creates a demand for more

intelligent controllers. In order to fulfill the optimal control requirement, the controller should have the ability to adapt to the process thus leading to self-tuning or adaptive controllers.

In recent years, several attempts have been made to derive simple self-tuning controllers with varying degree of success. One of our objectives here is to examine the practicalty of implementing advanced control algorithms on minimum microprocessor based hardware which can form the basis of a low cost adaptive controller. This controller should work as a single stand alone controller or, as a building block in a distributed control system.

Looking into the world of self-tuning or adaptive controllers, one would be amazed by the wide choice of algorithms available. Each of the algorithms has its own specific area of applications. However, of the available ones only a few are of a general purpose nature. The main requirement of a suitable algorithm is not only its performance in terms of stability and convergence but also of its complexity in terms of practical application in a hard real time environment.

In order to get a feel for the practical application of algorithms developed, a small process was chosen in order to test the algorithms practically and examine their viability in actual applications. A very important reason for choosing a small process to try the application was that, in the past, due to the stringent cost constrain on these processes, it prohibited them

from using more advance control methods. If it can be shown that a microprocessor based controller can be built for such a process, it would open up new dimensions in the control of small processes as well as proving the controller's viability in bigger plants.

The process chosen was a fibre crimping plant as shown in Fig. (1.1). Fibre comes in from below, passes through two chambers and is heated up by the steam jet. The combined steam and fibre is then allowed to pass through a jet and to expand against a cone to produce the crimp. Crimped yarn is then allowed to come out of the jet in the form of a plug, and, to cool and set in free air. The speed of the yarn feed and take up are kept constant. The control action here is to vary the steam temperature input in order to keep the height of the plug column constant. This intrinsically simple process is quite interesting when studied closely. It has a varying time constant and a varying time delay as explained in chapter 2. Apart from its complex behaviour, the process is a typical example of small processes, where, due to its low overall cost it can not afford the modification of existing instrumentation and, the new microprocessor based controller should cost about the same as that of its original analogue controller.

In the course of developing a suitable control algorithm for the process, a device was required to record data from the plant as well as trying out the developed algorithm on the plant. A suvey was carried out at the start of this project in 1976 to

find a suitable device for these purpose. However, at that time the application of microprocessors in process control was only in its infancy. There were no devices in the market which would fit our requirement. Therefore, a microprocessor based control algorithm development and implementation system has to be designed from scratch.

In order to satisfy the design requirement of being able to perform on-line control, data logging (up to 9 channels of different sampling rate) and program development at the same time, while, the size of the whole system is still kept to the standard 19-inch card frame for easy transportation to the factory floor, a novel interrupt driven multi-tasking software and hardware architecture using several levels of interrupt was developed for this system.

The idea was to divide functions in the system such as data logging, on-line control, program debugging, data transmission, data reception, servicing other peripherals, etc., into different tasks. Each tasks is driven by an interrupt. The interrupts are divided into eight priority levels. Each active task is allocated a priority level according to its hard real-time requirement. The priority level of a task can be changed by another task or by itself according to the state of the program or the hard real-time requirements at that moment. For example, the on-line control function normally allocated the highest priority level for accurate sampling of plant output and control output. After this vital operation, its priority would be

changed to a lower level to allow other peripherals to interrupt while continuing its control calculation. Furthermore, a task can also be activated or suspended by another task according to software requirement or external stimulus. This form of multi-tasking arrangement was found to be superbly versatile in later experiments.

Since the development of this novel concept, other systems of similar design but of bigger size have emerged. An early example was the modular system base on the Z-80 processor designed by A.W. Winston and T.B. Smith in 1978 (Ref. 48). The advantage of using interrupt driven multi-tasking architecture on microprocessor systems was examined in the early days by others such as Chin-Hwa Lee in 1977 (Ref. 49) and Y.C. Chien in 1980 (Ref. 50).

Another novel concept in the design of this program development system was to squeeze two ergonomically designed active front panels within the confine of the 19-inch card frame. One of the front panels was used by the programmer while the other front panel was used by the control engineer. The reason for having the programmers panel was due to the lack of in house program development aid. This panel was to provide means for hand programming and debugging initial software. The idea of having a separate control engineers' panel was that when a newly developed control algorithm was going through its acceptance test, the system can be left on the shop floor with only the control engineers' panel accessible. This allowed control

engineers on duty to change the tuning parameters, switch between conventional 3-term controller and the new algorithm for comparison, and, display various other parameters without worrying about the danger of corrupting the system or controller software. The novel architecture of this system and its versatile features are described in Chapter 3.

The first step in designing an adaptive controller for the nylon crimping plant was to deduce a realistic model for the plant as described in chapter 2. In chapter 4, previous attempts to improve the controller were examined. Armed with the plant knowledge obtained from past efforts and simulation results from the new model, it was recognised that; due to the five discrete step output level sensor, the process was too non-linear to be controlled by normal self-tuning control algorithm. Most self-tuning control algorithm at start up would need at least hundreds of samples before converging to a reasonable value around the set point even for a linear plant (ref. 16, 27, 28). The nylon crimping plant with its five discrete step output level sensor would take a long time to (or in some cases never be able to) supply the required among of useful data. Therefore, a non-linear design approach was adopted to design a non-linear adaptive controller for the process. It was proved to have marked improvement over the previous proportional and integral controller in real plant trials as shown in chapter 5 (section 5.3.5).

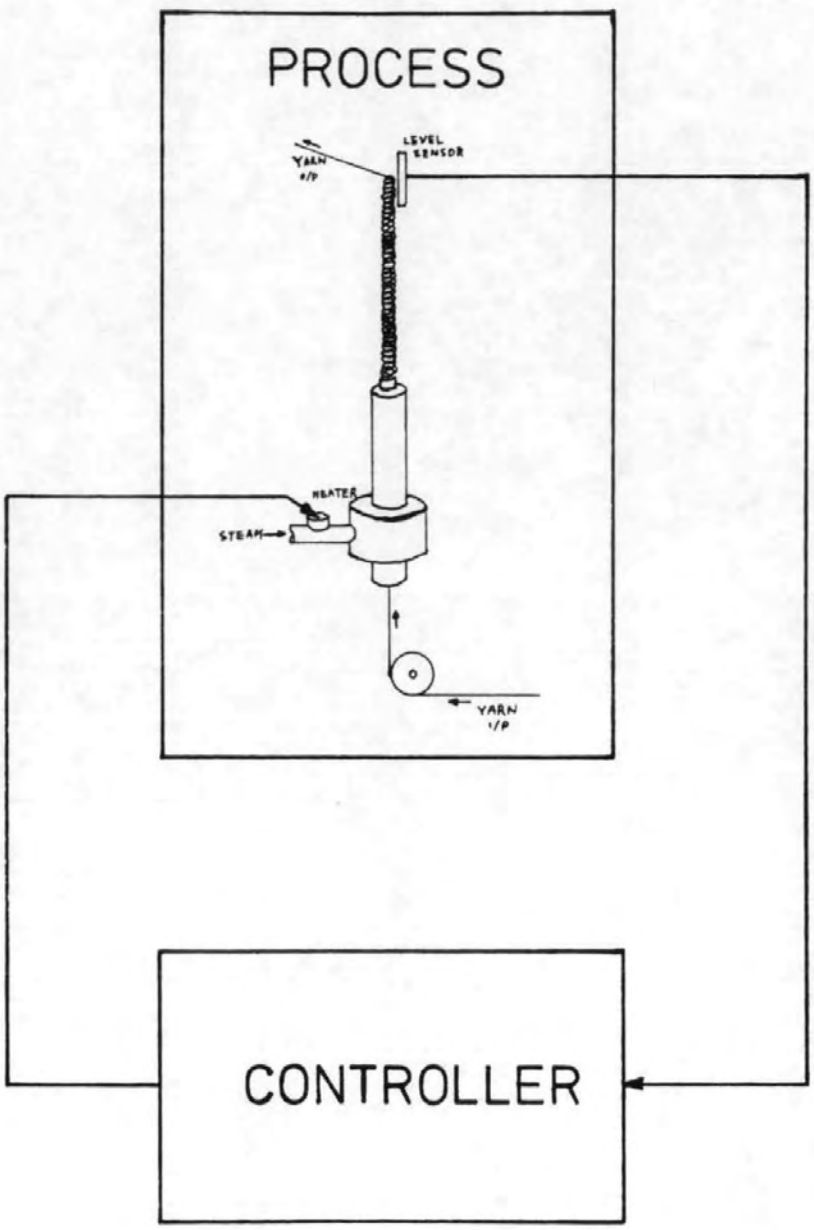
Chapter 5 also describes other experiments and on-line

identifications carried out on the plant with the aid of the Program Development/Implementation System. The versatility of the Program Development/Implementation System was proved to be extremely useful here.

Due to the format of the data collected, and, the difficult in converting it to a form which is acceptable to the main-frame computer in the college, it was decided to use the microprocessor based Program Development/Implementation System to analyse some of the data. Therefore, most of the analysis software was written in Intel 8080 assembler language.

When the control algorithm was fully developed, a dedicated microprocessor based controller using the Intel 8085 processor was designed and built for the plant. It was complete with plant interfaces, and, designed to have the same dimensions as the original analogue controller and directly plug compatible as well. It can be plugged straight into the existing rack for the analogue controller as a direct replacement.

Fig. 1.1 THE PROCESS CONTROL LOOP



2. THE PROCESS

In the field of self-adaptive control algorithms, there seem to be a very large selection of control algorithms. However, if we scrutinize them, only very few are suitable for general applications. If we take into account the hard real time requirement for process control and the limited computing power of a microprocessor the choice is even less. In order to examine these constraints practically, we chose a very convenient but interesting process to grasp the implication in practical application of advanced control algorithms in process control using microprocessors. The plant chosen also highlights the problem of small plants as described in the introduction that are badly in need of more advanced control algorithms while the economics does not justify the use of computers.

2.1 The Control Loop

The plant under study is an artifical fibre crimping plant as shown in the schematic diagram Fig 2.1. This plant was developed by John Heathcoat & Co. Ltd. It is an unique process using a combination of fluid jet and stuffer tube texturing. Yarn is sucked in from the bottom by the steam jet and its velocity maintained constant by a feeding roller. Inside the jet, the temperature of the yarn is increased to just below melting point by superheated steam. The yarn and the steam then pass through a series of jets and expansion chambers so that the yarn is being torn open and twisted. Finally, the yarn is forced into

a vertical tube to form a plug and pushes its way out to the open air, cooled down to sustain permanent crimp. The plug pushes its way up a guide with more crimped yarn forming at the bottom. Then crimp yarn is finally drawn off at constant speed.

It was found from experiment that with input and output roller speed kept constant the height of the plug changes according to the temperature of the input steam (higher temperature - shorter plug length). Furthermore, the amount of crimp (as indicated by dye take up) is proportional to the plug length. Hence, in order to maintain a constant amount of crimp, the plug length has to be kept constant by varying the temperature of input steam.

By theoretical realisation and practical experiment the plug length is found to be mainly dependon two factors: The temperature of the input steam and, the internal state of the yarn. The internal state of the yarn is further dependent on two major factors. First the water content in the yarn and secondly, the degree of relaxation after drawing. After drawing, yarn is stored in form of reels. Yarn on the outside of the reel tends to relax more than those in the middle which is restricted by the bobbin. Furthermore, yarn on the outside tends to be affected by the atmosphere more than those on the inside. The change of steam temperature in order to compensate for the different properties of the yarn to attain constant crimp during a typical run is shown in Figure 2.2. Since we can control the temperature of the steam but not the state of the yarn, we shall consider the temperature of the steam to be the control input to the plant and the change of state of the yarn to be the disturbance.

The control loop now becomes a single input and single output system. The plant output being the plug level and the control input being the power input to the heater. A schematic diagram of the control loop is as shown in Figure 2.3. The disturbances to the plant are fluctuation of input steam temperature and the changes in the internal state of the yarn as explained earlier.

2.2 System Modelling

There are usually several ways to model a plant. It is inevitable that assumptions are made at some stages. Therefore, it is essential to look at different modelling approaches and perform simulations to see whether the plant dynamics agree with the model.

2.2.1 Previous modelling approach

A previous attempt to model the system is described as follows: First consider the input and output of yarn to the plant. From the set up of the plant, the tension and velocity of the yarn at input and output are held at constant. Therefore, we can assume that their mass flow rates are equal.

i.e. $$\dot{m}_i = \dot{m}_o \qquad (2.2.1)$$

$\dot{m}_i$ = input mass flow rate.

$\dot{m}_o$ = output mass flow rate.

From the above, assume that the time taken for material to travel the length of the plug is constant and is given by

$$\tau = \frac{M}{\dot{m}_i} \qquad (2.2.2)$$

M = mass of material in the plug.

The rate of change of the plug height L is equal to the difference between the rates at which yarn is added to the base and subtracted from the top:-

$$\frac{dL}{dt} = \dot{V}_b(t) - \dot{V}_t(t) = \dot{m}_i\, q_b(t) - \dot{m}_o\, q_t(t) \qquad (2.2.3)$$

$V_b(t)$ = volume of yarn added to base of plug.

$v_t(t)$ = volume of yarn subtracted from top of plug.

$q_b(t)$ = specific volume at base of plug.

$q_t(t)$ = specific volume at top of plug.

Substituting the above equation with equation (2.2.1) we have

$$\frac{dL}{dt} = \dot{m}_i\, (q_b(t) - q_t(t)) \qquad (2.2.4)$$

As the yarn moves from the bottom to the top of the plug the specific volume of yarn at top of the plug is equal to that at the base τ seconds previously. Hence,

$$\frac{dL}{dt} = \dot{m}_i\, (q_b(t) - q_b(t - \tau)) \qquad (2.2.5)$$

Taking the Laplace Transform:-

$$s\, L(s) = \dot{m}_i\, (q_b(s) - q_b(s)\, e^{-\tau s}) \qquad (2.2.6)$$

Assume that q_b is inversely proportional to temperature of steam.

$$\therefore\ L(s) = -\frac{K_1}{1 + s\theta} \left(\frac{1 - e^{-\tau s}}{s}\right) U(s) \qquad (2.2.7)$$

Let N(s) be the disturbance due to the change of states of the yarn. The transfer function becomes:-

$$L(s) = \frac{-K_1}{1 + s\theta}\left(\frac{1 - e^{-\tau s}}{s}\right) U(s) - K_2\left(\frac{1 - e^{-\tau s}}{s}\right) N(s) \qquad (2.2.8)$$

Where K_1 , K_2 = constants

$U(s)$ = plant input (power input to heater)

θ = heater time constant

For easy analysis of the control loop in computer control applications. The Z-transform of the above transfer function is as follows:-

Assuming that the time delay is an integer multiple of the sampling rate,

$$G(Z) = \frac{K Z^{-1} (1 + Z^{-1} + Z^{-2} + ... + Z^{-(m-1)}) (p + q Z^{-1})}{1 - e^{-\frac{T}{\theta}} Z^{-1}} \qquad (2.2.9)$$

Where $p = T - \theta(1 - e^{-\frac{T}{\theta}})$, $\tau = m T$,

$q = \theta (1 - e^{-\frac{T}{\theta}}) - T e^{-\frac{T}{\theta}}$, T = sampling period.

2.2.2 Recent modelling approach

Further realisation of the plant have shown that the validity of the model derived in section 2.2.1 is questionable. From equation 2.2.5 for any change of $q_b(t)$, the rate of change of L is equal to zero after τ seconds. That is to say, the system is stable in open loop condition. However, on examination of the plant, the open loop condition is unstable. This error in modelling arises from the assumption that the input and output mass flow rate is equal, which is not true at all. Since the output yarn speed and tension is kept constant the mass flow rate is dependent on the amount of

crimp on the yarn. More crimp on the yarn increases the mass per unit length of the yarn and hence increases the output mass flow rate.

The main disturbance to the system is the relaxation of the yarn after drawing. Since the input yarn travels at a constant speed, any relaxation of the yarn after drawing would increase its mass/unit length which would result in an increase in mass flow rate.

An alternative approach to modelling of the process is as follows:-

Define: G_i = linear density of non-crimped yarn (mass/unit length)

G_o = linear density of crimped yarn

Q_t = linear density of plug at top of column

Q_b = linear density of plug at bottom of column

C = temperature of steam

V_i = velocity of non-crimped yarn (constant)

V_o = velocity of crimped yarn (constant)

L = plug length

$\dot{m}_i$ = input mass flow rate

$\dot{m}_o$ = output mass flow rate

E_i = energy input to super heater

The increase in plug length is due to yarn added to the bottom of the plug,

$$\frac{dL_b}{dt} = \frac{\dot{m}_i}{Q_b} \qquad (2.2.10)$$

The decrease in plug length is due to yarn taken away from the top of the plug,

$$\frac{dL_t}{dt} = \frac{\dot{m}_o}{Q_t} \qquad (2.2.11)$$

Hence the change in plug length

$$\frac{dL}{dt} = \frac{\dot{m}_i}{Q_b} - \frac{\dot{m}_o}{Q_t} \qquad (2.2.12)$$

Let τ be the time for the yarn to travel from the bottom of the plug to the top.

$$\therefore \quad \frac{dL}{dt} = \frac{\dot{m}_i}{Q_b(t)} - \frac{\dot{m}_o}{Q_b(t - \tau)} \qquad (2.2.13)$$

Now $\dot{m}_o = G_o V_o$

$\dot{m}_i = G_i V_i$

Where V_o & V_i are fixed by the operator and remain constant.

The amount of crimp on the output yarn is dependent on the steam temperature C(t) in the jet. Assume that this relationship is piece wise linear around the set point,

$$\therefore \quad G_o = K_G C(t - \tau) + A_G \qquad (2.2.14)$$

Where K_G and A_G are constants

Also assume that the relationship between Q_b and jet temperature C is piece wise linear around the set point,

$$\therefore \quad Q_b = K_Q C(t) + A_Q \qquad (2.2.15)$$

Where K_Q and A_Q are constants

$$\therefore \quad \frac{dL}{dt} = \frac{G_i V_i}{K_Q C(t) + A_Q} - \frac{V_o (K_G C(t - \tau) + A_G)}{K_Q C(t - \tau) + A_Q}$$

$$= \frac{G_i V_i}{K_Q C(t) + A_Q} - \frac{K_o C(t - \tau) + A_o}{K_Q C(t - \tau) + A_Q} \qquad (2.2.16)$$

Where $K_o = V_o K_G$

$A_o = A_G V_o$

τ = Time for yarn to travel from bottom to top of plug.

$$\tau = \frac{L}{\dot{L}_b(\text{Average})} = L \frac{(K_Q \int_{t-\tau}^{t} C(t)\, dt) \frac{1}{\tau} + A_Q}{G_i V_i} \qquad (2.2.17)$$

Where $\dot{L}_b$ = Velocity of yarn travelling up the plug.

In order to simplify equation (2.2.16) assume that change of Q with respect to temperature is very small and can be ignored which is true in practise.

$$\therefore \frac{dL}{dt} = G_i B_i - B_o C(t - \tau) - A_e \qquad (2.2.18)$$

Where $B_i = \frac{V_i}{Q_b}$, $B_o = \frac{K_o}{Q_b}$, $A_e = \frac{A_o}{Q_b}$.

$$\therefore s L(s) = G_i B_i - B_o C(s) e^{-\tau s} - A_e$$

$$= G_i B_i - B_o \frac{K_e}{1 + \theta s} e^{-\tau s} E_i - A_e$$

$$= B_R \frac{e^{-\tau s}}{1 + \theta s} E_i + B_d (\zeta + \psi) e^{-\tau s}$$

Where $B_R = - B_o K_e$ = Constant,

B_d and ψ = Constant,

ζ = Disturbance,

$C(s) = \frac{K_e}{1 + \theta s} E_i$,

$$\therefore L(s) = B_R \frac{e^{-\tau s}}{s (1 + \theta s)} E_i + \frac{B_d}{s} (\zeta + \psi) \qquad (2.2.19)$$

From equation (2.2.19) it has been shown that the open loop transfer function of the system can be simplified to a second order system with variable time delay. Now, this simplified version of the model is stable while the observed open loop system is unstable. This contradiction can be explained as follows: If the $K_Q\ C(t) + A_Q$ terms in (2.2.16) is included in the model, the model would be unstable. Although (2.2.16) gives a more accurate representation of the plant it gives a very complex open loop transfer function. The way to get round this problem while still retain a simple model of the system is that, we shall take the unstable term as drift which can be overcome in closed loop by a pure integrator on the controller. The system is then further compensated as a delayed second order system.

In order to further validate this model, some model fitting and identification results are presented and discussed in Chapter 5.

For easy analysis by computer the z-transform of the simplified open loop transfer function is as follows:
Assume that the time delay is an integer multiple of the sampling period (T) the transfer function becomes:

$$L(Z) = B_R \frac{z(1 - e^{\frac{-T}{\theta}})}{(z - 1)(z - e^{\frac{-T}{\theta}})} z^{-n} E_i(z) \qquad (2.2.20)$$

Where time delay $\tau = nT$

From the z-transform of the system model in equation (2.2.20) the discrete-time state-space model of the system can be derived for later analysis by modern control theory:

Let $L(z) = x_1(k)$, $x_1(k + 1) = x_2(k)$

From equation (2.2.20)

$$L(z)\, z^2 - L(z)\, z(1 + e^{-\frac{T}{\theta}}) + L(z)\, e^{-\frac{T}{\theta}} = z^{-(n-1)}\, B_R(1 - e^{-\frac{T}{\theta}})\, E_i(z)$$

Hence the discrete-time state-space model of the system becomes:

$$\begin{bmatrix} x_1(k+1) \\ \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \\ -e^{-\frac{T}{\theta}} & 1 + e^{-\frac{T}{\theta}} \end{bmatrix} \begin{bmatrix} x_1(k) \\ \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} B_R(1 - e^{-\frac{T}{\theta}})\, E_i(k - n + 1) + Q\,\zeta$$

(2.2.21)

$$L(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ \\ x_2(k) \end{bmatrix}$$

Where ζ is the disturbance

k is any integer

Therefore, matrix representation of the open loop system is:

$$X(k+1) = A\,X(k) + B\,E_i(k - n + 1)$$

$$L(k) = D\,X(k) \qquad (2.2.22)$$

In order to present the model in true state-space format we must eliminate $E_i(k - n + 1)$ in equation (2.2.22), which we do by defining a new state $v_1(k) = E_i(k - n + 1)$, $v_2(k) = E_i(k - n + 2)$,... ....., $v_{n-1}(k) = E_i(k - 1)$.

Substituting the new state into (2.2.22) we have:-

PLEASE SEE NEXT PAGE

$$
\begin{bmatrix} x_1(k+1) \\ x_2(k+2) \\ v_1(k+1) \\ v_2(k+1) \\ \cdot \\ \cdot \\ \cdot \\ v_{n-2}(k+1) \\ v_{n-1}(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ -e^{\frac{-T}{\theta}} & 1+e^{\frac{-T}{\theta}} & B_R(1-e^{\frac{-T}{\theta}}) & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \cdots & 0 \\ \cdot & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ v_1(k) \\ v_2(k) \\ \cdot \\ \cdot \\ \cdot \\ v_{n-2}(k) \\ v_{n-1}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix} E_i(k)
$$

$$
L(k) = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ v_1(k) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_{n-1}(k) \end{bmatrix}
$$

(2.2.23)

Fig. 2.1 Schematic Diagram of the Nylon Crimping Plant.

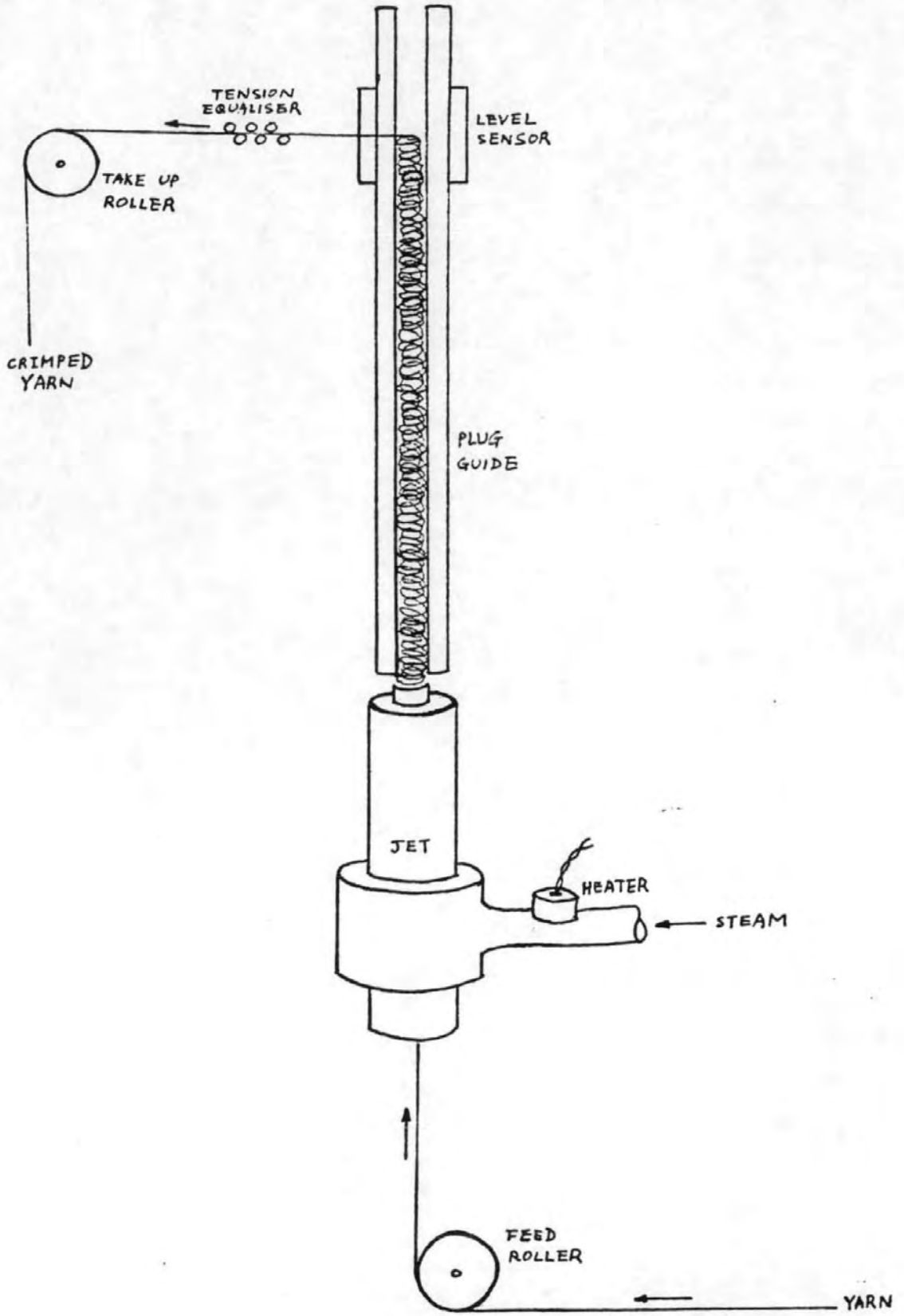


Fig. 2.2 Steam Temperature During A Typical Run For A Whole Bob Of Yarn.

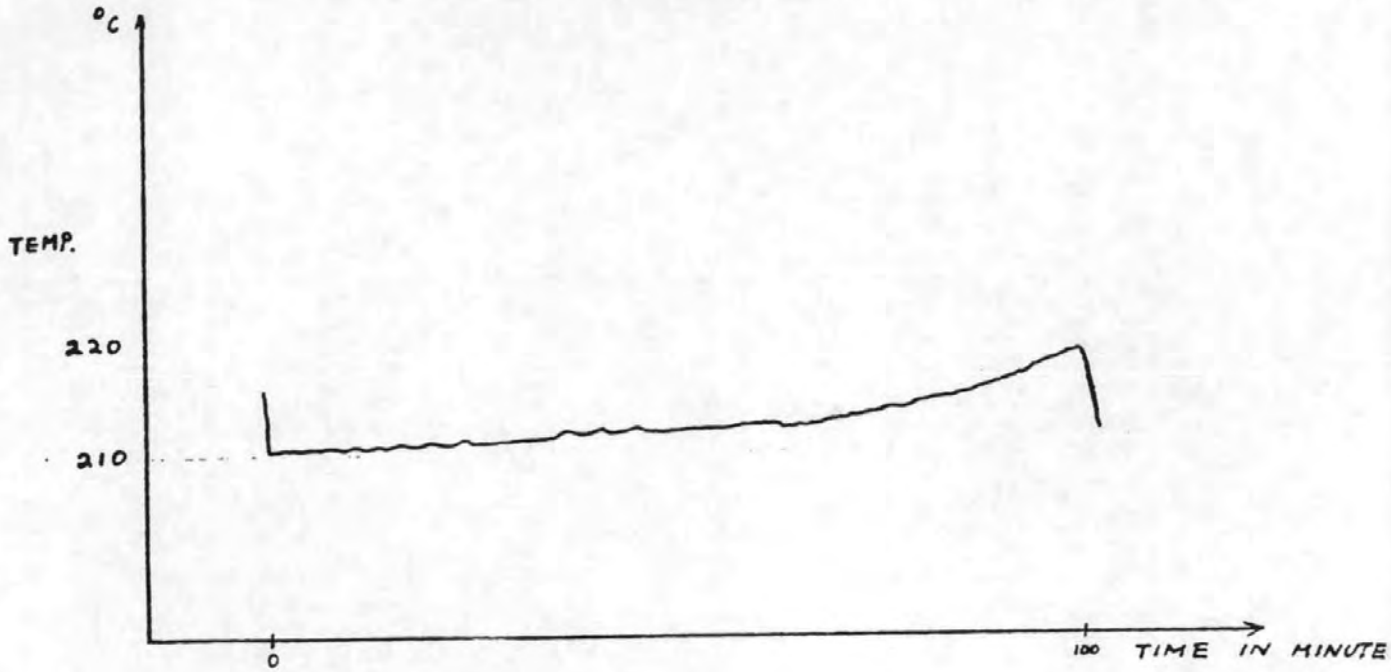
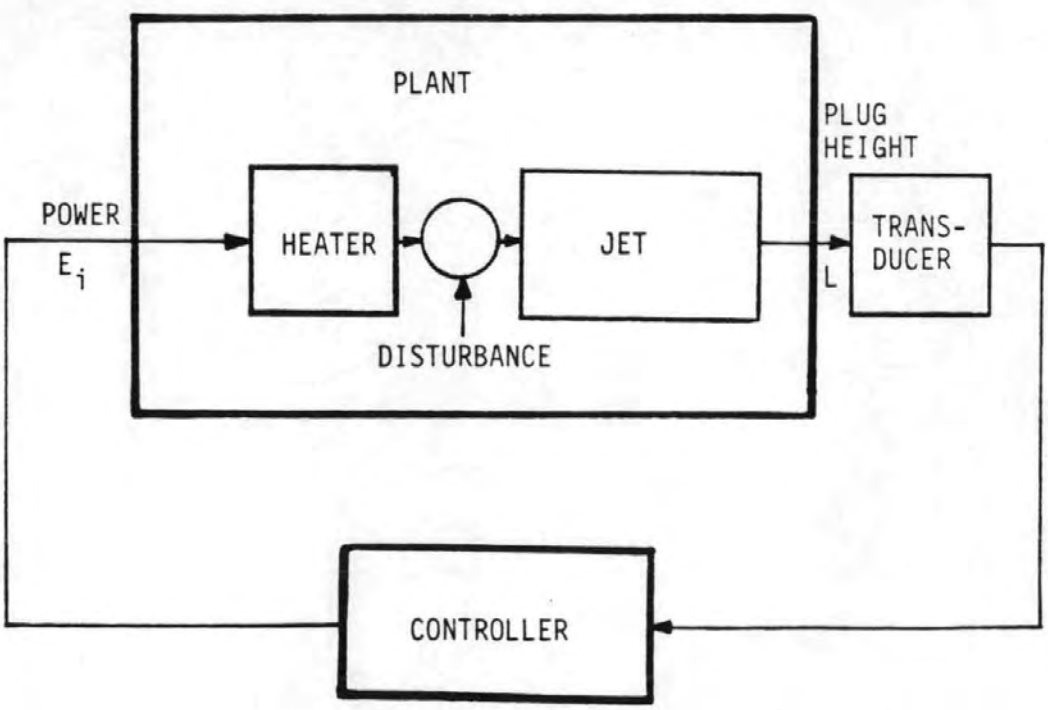


Fig. 2.3 Schematic Diagram of Control Loop



3. THE ALGORITHM DEVELOPMENT/IMPLEMENTATION SYSTEM

3.1 Introduction

In order to test and develop a control algorithm on an industrial plant inside a factory a flexible device is required to implement the algorithm, log its performance and allow the algorithm to be modified. At the same time, the device should be able to stablise the plant even when there is a fault in the algorithm under development. In some cases where the plant is already controlled by a computer, the above mentioned device may not be necessary. However, it is still very useful in some localised control loops.

Furthermore, it is vital to test the algorithm on the same processor as that to be used on the target controller so that the computation time and robustness of the algorithm on a particular processor can be fully appreciated.

Searching around, there was no such device on the market at the beginning of this project which would satisfy the requirement stated above. The simple fact was that, at that time (1976) eight bit microprocessors had only just come on the market and, the decision was to use an eight bit processor. Therefore, an Algorithm Development/ Implementation System had to be developed before the control algorithm could be tested on the actual plant.

3.2 Design Philosophy

As explained in section 3.1 there was a need for a device to test the control algorithm on site and there was nothing suitable on the market. The idea was to develop a general purpose device so that it would serve the requirements of this project as well as being a useful tool for future research and teaching purposes. The system developed might have been sold as an instrument if enough interest had been generated.

With the above parameters in mind the initial specifications of the Algorithm Development/Implementation System was drawn up as follows:

(1) A hardwire bootstrap system for loading and debugging initial software - due to the absence of software development system at that time.

(2) A control engineer's front panel - so that when the algorithm is on proofing trial, the device can be left in the factory for control engineers to alter the tuning and set points, but unable to alter or destroy the program.

(3) Interrupt driven multi-tasking software and hardware structure - so that it allows manipulation of the control algorithm even when the device is controlling the plant.

(4) Programmable multi rate, multi channel data logger - for logging the performance of the control algorithm under trial, identification of the plant and look at inter-sample response of the system.

(5) An incremental 3-term controller - so that the plant can be stabilised roughly by switching over to 3-term control when the algorithm under develop fails to stabilise the plant.

(6) Physical size of the device must be kept to minimum - for easy transportation.

Having defined the objective an Algorithm Development/Implementation System was designed and built around the above initial specifications.

3.3 System Functions

3.3.1 General Description

The microprocessor system was designed to perform two functions: program development and on-line control. The system is an interrupt driven multitasking system with the level of priority chosen to give the optimum saving of computing time. Therefore, at most times the computer would work as if all the tasks are performed in parallel.

The controller of the system can accept binary or analogue input and output.

There are two communication interfaces to the system; TX1 and TX2. TX1 is for data logging while TX2 is for data logging and program development.

3.3.2 The front panel

The front panel of the system is as shown in Fig. 3.1. It is divided into 2 parts. The control engineer uses the panel on the right and the programmer uses the panel on the left. On the control engineer's

panel, all the inputs and displays are in the form of four digit decimal fraction with a 2 digit exponential. While on the programmer's panel, all the inputs and displays are in Hexadecimal.

3.3.2.1 Programmer's panel

The display A & B are for address and data display respectively. C & D are Hex decimal thumb wheel switches for inputting the same. The switch functions are self-explanatory.

To examine the content of an address, set up the address using the thumb wheel, then switch (4) to 'DATA' and press (5) to 'EXAMINE'. If you want to load data onto that address, you set up the data on (D) and press (5) to 'LOAD'.

Switch (3) if on 'BREAKPOINT' when (2) is on 'STOP' and (6) is pressed, the program would execute from the present instruction and stop at the address on (C). If (3) is on 'GO TO' when (6) is pressed, the program would jump to address on (C) immediately.

3.3.2.2 Control engineer's panel

The display (E) and (F) are the decimal part and exponential part of a parameter in the controller respectively, whose address is shown on (J). When the controller is running, this display is automatically up dated between samples. When the address is being set up on (K) and the display switch pressed, the parameter would be displayed

on (E) and (F) immediately. When the enter switch is pressed the number on (G) and (H) would be loaded into the address shown on (K). The addresses 0, 1 and 2 are timer output and input respectively. All addresses from 3 to 99 can be assigned by the programmer for any purpose.

The sampling period of the controller can be programmed on this front panel from 40 ms to 100 s. in any sequence. The programming procedure can be found in Appendix 7. The data logger can be programmed to a number of patterns of logging as well. It can log up to 9 parameters per sample and can be programed to delay data logging for a number of samples, so that we can use fast sampling rate for control and slower rate for logging. The programming procedure can be found in Appendix 7.

3.3.3 The teletype monitor

When the system has completed all other tasks, it will return to the teletype monitor. The teletype monitor can only be accessed through communication interface 2 (TX2). When the system was reset the baud rate is set to 110. This baud rate can be altered by software. The commands for the teletype monitor are quite conventional except for the trace (T) command which is a very powerful command for program debugging.

The different commands for the teletype monitor are described in Appendix 3. A summary of the commands are as follows:-

Summary of Commands

C	:	COPY a block of data from one location to another.
D	:	DUMP data onto paper tape.
E	:	ENTER data or program into memory.
G	:	GO to start of program execution.
L	:	LOAD data from paper tape.
M	:	MODIFY data in the memory.
T	:	TRACE error in the program.
W	:	WRITE memory content on display(TTY).
X	:	Change registers, flags and stacks.
.	:	Exit from commands.

3.4 System Hardware

3.4.1 General Description

The system architecture is a very conventional uni-bus design, its schematic diagram is as shown in Fig. 3.4.1.

The D/A and A/D converter board is optional. If analogue input and output are not required, it should be unplugged from the connector. The system was designed for easy maintenance, low cost and space saving.

3.4.2 The front panel

The front panel of the system is as shown in Fig. 3.1. All the displays are 7 segment Hexadecimal displays. The thumb switches on the Programmers' Panel (left) are Hexadecimal while those on the Control Engineers' Panel (right) are decimal. All the control logics are situated behind the front panel for space saving. All chips are easily accessable except the 7 segment displays. The circuit of the front panel can be found in Fig. 3.4.3.

3.4.3 The processor board

The Processor Board houses the 8080A-1 processor, the Clock Generator and Driver (8224), the System Controller and Bus Driver (8228), the Address Bus Buffer (8212), the Address Decoder (8205), the Priority Interrupt Control Unit (8214) and four 1K x 8 EPROM (8708-2). There are some hard-wired delay circuits in order to allow

the processor to work with some slower devices. This board is self-contained, it is functional even without the support of other boards. This makes fault finding easier because faults can easily be traced to individual boards. The circuit can be found in Fig. 3.4.4.

3.4.4 The interface board

The interface board houses all the interface to the outside world except the D/A and A/D converters. It includes one programmable Timer(8253), two Programmable Communication Interface (8251), two Parallel Input Latches and two Parallel Output Latches, a couple of Dividers and a few Optical Isolators to the outside world. There are three Programmable Timers in the chip 8253. Two of them are used for controlling the baud rate of the two Communication Interfaces. The third one is for controlling the sampling rate of the controller. All timers here use the system clock. Because they only work up to 2 MHz, while the system clock is 3 MHz, the system clock is first divided by two before feeding to the baud rate generators. For the sampling rate generator the clock was divided down by thirty thousand before feeding into the timer. Since some of the devices here work slower than the processor, a delay request is organised to slow down the Processor when these devices are activated. All outputs and inputs from this board to outside world are buffered with optical isolators and in the form of 20 mA current loop to minimise transmission noise. The circuit diagram of this board can be found in Fig. 3.4.5.

3.4.5. The D/A and A/D Board

This Board performs the digital to analogue and analogue to digital function. Incoming analogue voltage is first sampled and held for a short period for the A/D converter to convert the analogue signal to a 3 digit B.C.D. number. The number is then latched onto buffers ready for the Processor to read in. The digital output from the Processor is also in form of 3 digit B.C.D. and latched onto the input of the D/A converted and an analogue signal of the prescribed magnitude shall remain on the output of the D/A until another change comes along. The input and output of the D/A and A/D to the outside world are protected by double zener diode and fuse circuits. The circuit diagram of this board can be found in Fig. 3.4.6.

3.4.6 The RAM Board

There are four 1K x 8 R.A.M. Board in this device. One 1K x 8 is used by system as scratch pad and table storage. The other three 1K x 8 RAM are for user programs and 3-term controller. It was decided that 3K is quite enough for the normal control algorithms, because algorithms requiring a long program length may not meet the hard real time requirement of the sampling rate. However, if a larger storage is required the memory capacity can be increased. Each RAM board here is self-contained, completed with output buffers and stand-by batteries. The batteries are quite large although only supplying the memories for 27 hours. This was because no low power version of RAMs with an access time less than 330 ns were available at that time. The board also disables 'WRITE' at power down in order to prevent it being wiped out

by the misadventure of the Processor at switch off. The circuit diagram of this board can be found in Fig. 3.4.7.

3.5 System Software

3.5.1 General description

The software of this device can mainly be divided into the Operating System, Teletype Monitor, Controller and Data Logger, and some Arithmetic Routines. The Operating System, Teletype Monitor, Controller and Data Logger occupied 3K of EPROM with 1K RAM for scratch pad and tables. The Arithmetic Routines occupied another 1K of EPROM giving a total of 4K of EPROMS. The memory map of this system can be found in Appendix 2. One unique feature of this system is the interrupt structure which shall be explained in the next section.

3.5.2 The interrupt structure

The Priority interrupt control unit (8214) chip provides 8 levels of priority. They are wired to different functions as follows:

R_7 - Reset Initialisation

R_6 - Timer Interrupt

R_5 - Receiver ready communication interface 1 (TX1)

R_4 - Transmission ready communication interface 1 (TX1)

R_3 - Receiver ready communication interface 2 (TX2)

R_2 - Transmission ready communication interface 2 (TX2)

R_1 - Servicing input/output from front panel

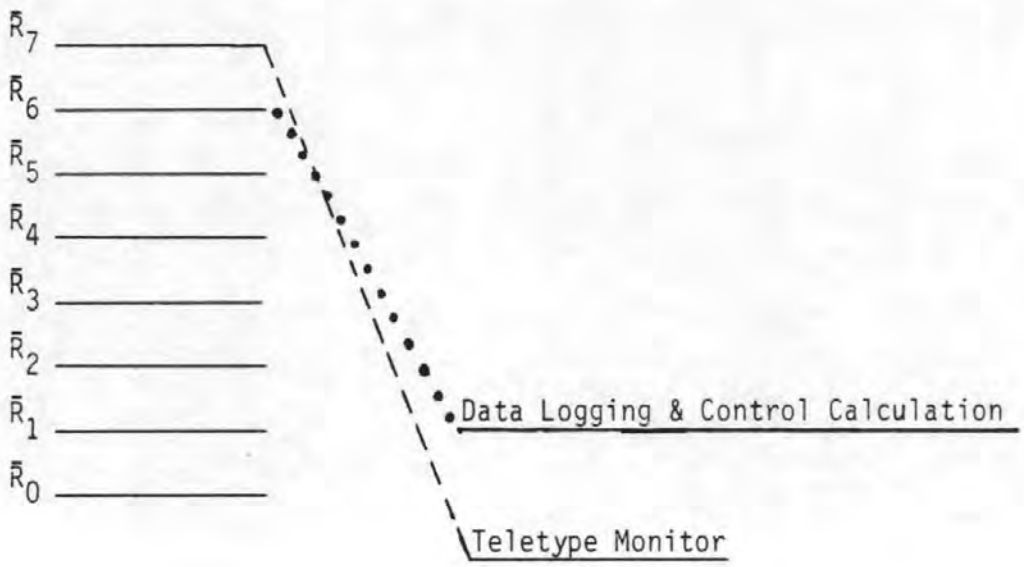
R_0 - From 'Execute' switch on front panel.

Apart from the above hardwired interrupt levels, some of the interrupt service routines change their level of interrupt during their course (see Fig. 3.5.1). The reset initialisation (R_7) is one of them. When R_7 is initialised (i.e. the system is being reset) it goes through a series of initialisation routines resetting all buffers, programmable timers and interfaces. When these are done it then drops its priority to the lowest level to allow all interrupts to come in before settling in the teletype monitor program.

The timer interrupt (R_6) is another interrupt service routine which changes its interrupt level during its course. After the timer interrupt has serviced the controller input and output it then drops its priority to R_1 to run the data logger and user program. Since no variable should be changed during data logging and control calculation, interrupt level R_1 and downward should be inhibited. Whereas, the hard real time requirement for reception and transmission are more stringent than the data logging and calculations, besides, they also do not occupy a lot of computing time. Hence, they should be allowed to come in so that input and output can continue. It may seem straight forward to change the level of priority during the course of the service routines, which simply means reprogramming the priority interrupt control unit. However, if one looks closer, or just considers the example of interrupt service routine for R_5 being interrupted by R_6 and during its course of action the priority is changed from R_6 to R_1. Then, although the program is running at the priority of R_1 it would not service R_5 until it completes all its tasks. Hence, if some other interrupt shows up during this period, it will be serviced first disrupting the normal running of the priority scheduling.

This difficulty is overcome by detaching the lower priority part of the routine from the higher one. when R_6 has been serviced it would set a flag to signify that data logging and calculation starts. Then it goes on to check whether other higher priority (R_2 upwards) routines have been completed. If not, they would be completed before the second part of R_6 could continue. When the data logging and calculation has been completed the flag would be reset.

Fig. 3.5.1



3.5.3 The operating system and teletype monitor

The Operating System basically consists of eight interrupt service routines. The highest priority one (R_7) is the System Reset and Initialisation. It sets up the stack pointer, resets all the buffers, reprograms the Programmable Timers and the Communication Interfaces. Then it drops its interrupt priority level to the lowest and enters the Teletype Monitor. The Teletype Monitor is made up of three groups of routines. The main program is the Command Recogniser.

It identifies the inputted command and jumps to the required sub-routines for servicing the command. The second group of routines are the Command Implementation subroutines. They do the work of implementing a specific command. The third group of routines are Utility Subroutines. They constitute the nuts and bolts of the Command Implementation Subroutines and do all the repetitive work for them.

The second highest priority level (R_6) is the timer interrupt. It is assigned to this level because it is the most important. It services the input and output of the controller which cannot wait. It also services the data logger and control calculations at a lower priority level. The next four interrupt levels (R_5, R_4, R_3, R_2)are for the Communication Interface. There are output buffers of size 98 bytes for each interface so that the program would not be delayed by output devices. When output is required it simply dumps the output on the buffer where it is automatically taken care of by the output devices.

The second lowest priority level (R_1) is for servicing input/ outputs from the front panel. When the display or enter button on the front panel is pressed, this interrupt is initialised. It first reads the constant address from the front panel and finds out whether it is 'Data Entry or Display'. Then the required function for this address is performed. The lowest priority interrupt level (R_0) connected to the priority interrupt controller is from the 'Execute' button on the front panel. When this interrupt is activated it will read the address set up on the front panel and push it onto the program

counter (i.e. jump to the address specified on the front panel).

3.5.4 The floating point arithmetic routines

The Floating Point Arithmetic Routines consists of three routines: The Addition and Subtraction Subroutine, the Multiplication subroutine and the Division Subroutine. The inputs to the routines are in the form of decimal fraction and its exponent. The decimal fraction part is in the form of a 4 digit B.C.D. (binary coded decimal) number. The sign of the number, the exponent and its sign are all combined to form an 8 bit word with the first bit (M.S.B.) for the sign of the number, the second bit for the sign of the exponent and the remaining 6 bits for the exponent in pure binary. The logical '1' is defined as positive for the sign bit. The arrangement of the operator and operand in the registers are as follows:

Registers B and C are for decimal part of the Operator.

Registers D and E are for decimal part of the Operand

Registers H and L are for the signs and exponent of the Operator and operand respectively.

One important point for these routines is that there should be no zero at the most significant digit of the decimal part on both the operator and operand.

These routines will work to an accuracy of 4 significant figures with round off at the least significant figure. They will work within the exponent range of ± 63. The listing of the program can be found in Appendix 6.

3.5.5 The binary arithmetic routines

The routines were written to simplify double byte binary arithmetics. Before the calling of these routines the operator should be placed on registers H and L. The results of the calculation would be resident in registers H and L. For the subtraction subroutine, if the output is negative, it will be in the form of 2's complement with both the sign and carry flag set. The listing of these routines can be found in Appendix 6.

3.5.6 The 3-term control algorithm

This program is normally resident at address 1CØØ. The reason for using an incremental algorithm for this machine is to ensure bumpless transfer. This enables the controller to be switched in to perform a degraded control in case the control algorithm under test fails. The control algorithm is as follows:-

$$\Delta C = K_I T_s E_t + K_p(E_t - E_{t-1}) + \frac{K_D}{T_s}(E_t - 2E_{t-1} + E_{t-2})$$

$$C_t = C_{t-1} + \Delta C$$

Where C_t is the controller output, E_t is the error signal and K_I, K_p, K_D are the integral gain, proportion gain and differential gain respectively. The derivation of this formula can be found in Appendix 12.

The constant address for K_p, K_I, K_D, E_t, E_{t-1}, E_{t-2} are 4, 5, 6, 7, 8, 9, respectively while the set point is at address 3.

The output and input of this controller can either be in B.C.D. for the A/D, D/A converter or in 8 bit pure binary. In the case of B.C.D.

input or output the exponent of the input or the output should be set to '0'. While, in the case of binary input or output the exponent of the corresponding addresses should be set to '1'. All settings for K_p, K_I, K_D should be in floating point numbers.

Fig. 3.1 Layout of the Front Panel.

PROGRAMMER'S PANEL

CONTROL ENGINEER'S PANEL

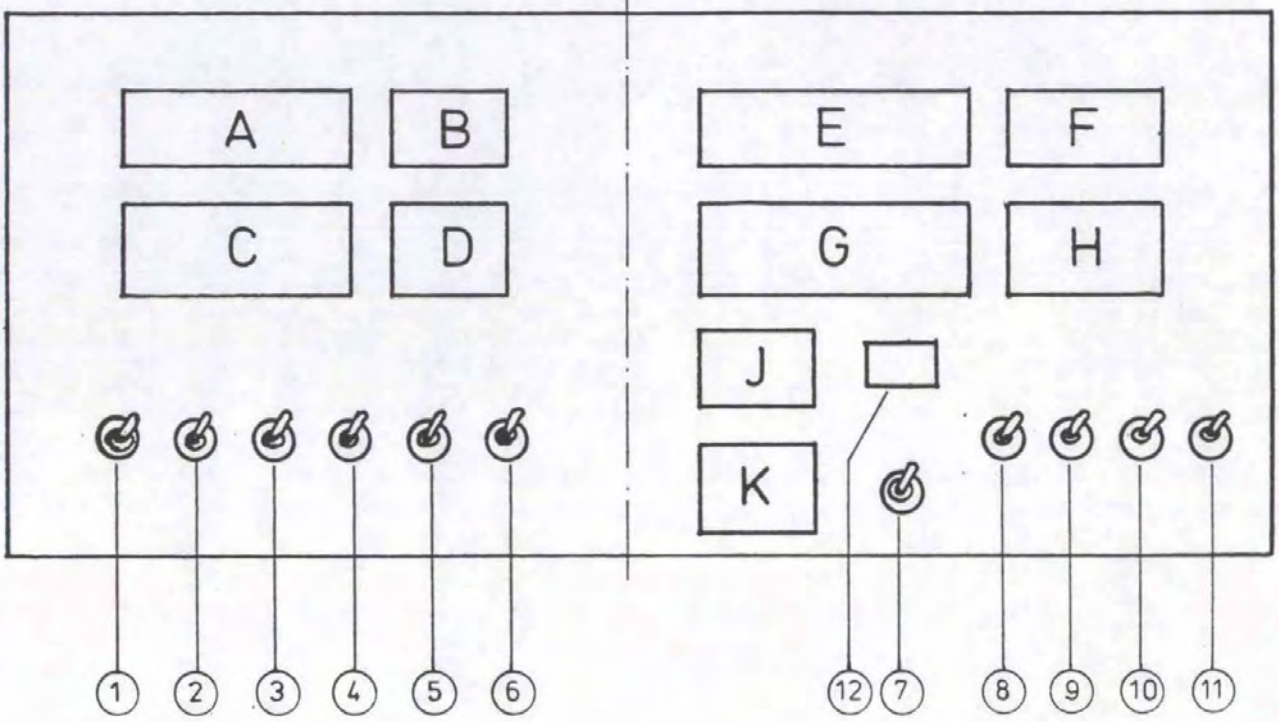


Fig. 3.4.1 Block Diagram of the 'Algorithm Development/ Implementation System'.

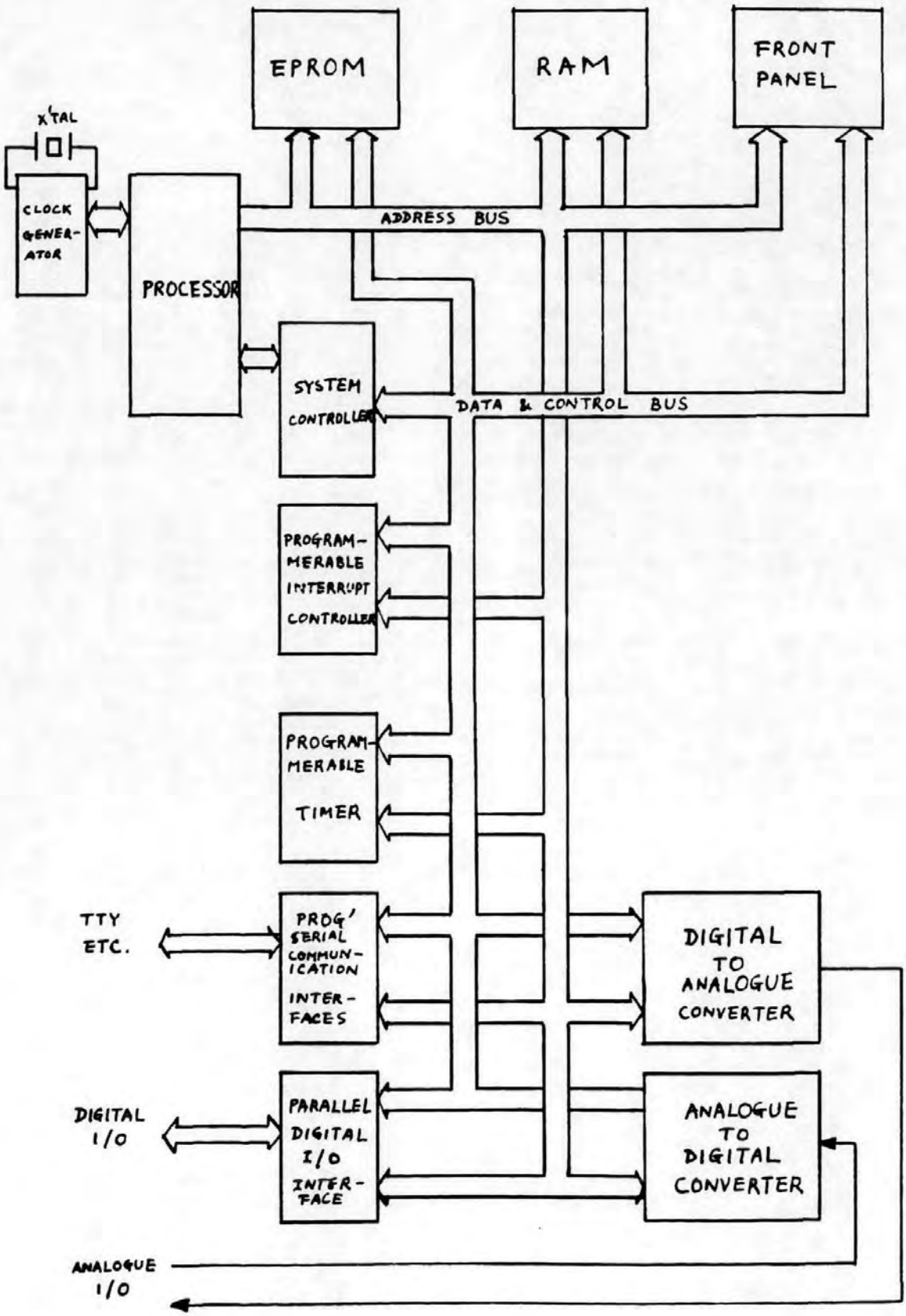
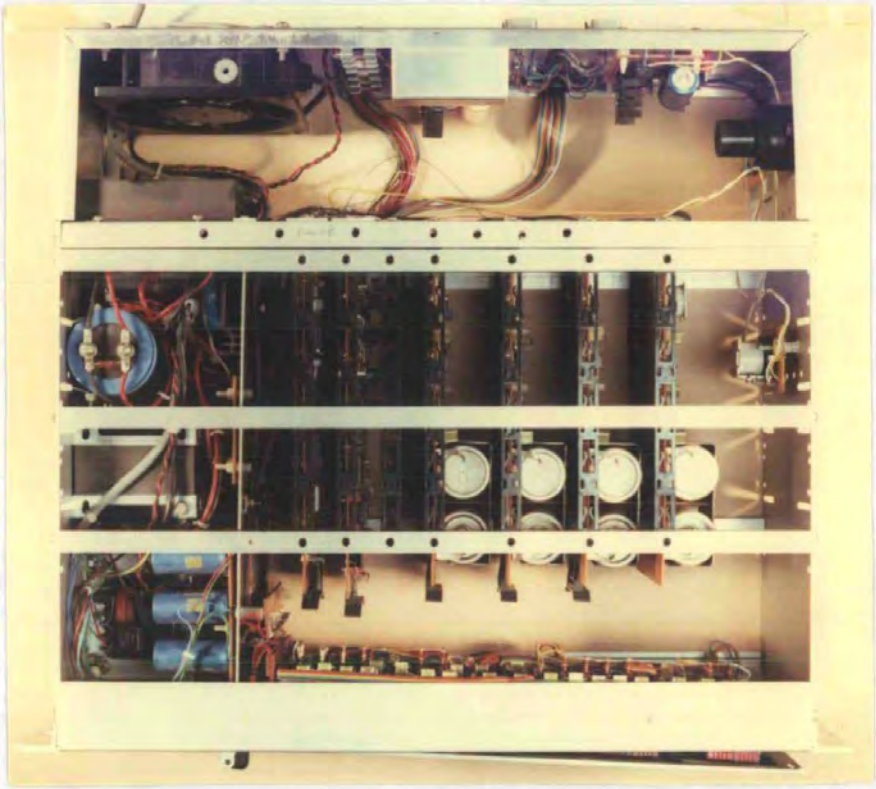
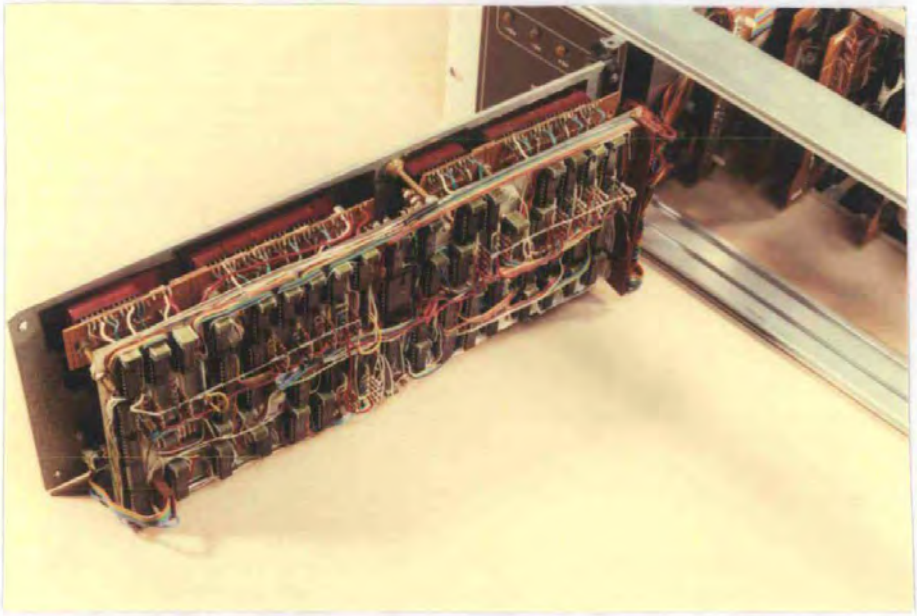


Fig. 3.4.2 Hardware Construction of the Algorithm Development/Implementation System.



INSIDE PLAN VIEW



INSIDE VIEW OF FRONT PANEL

FIG.3.4.3. THE FRONT PANEL

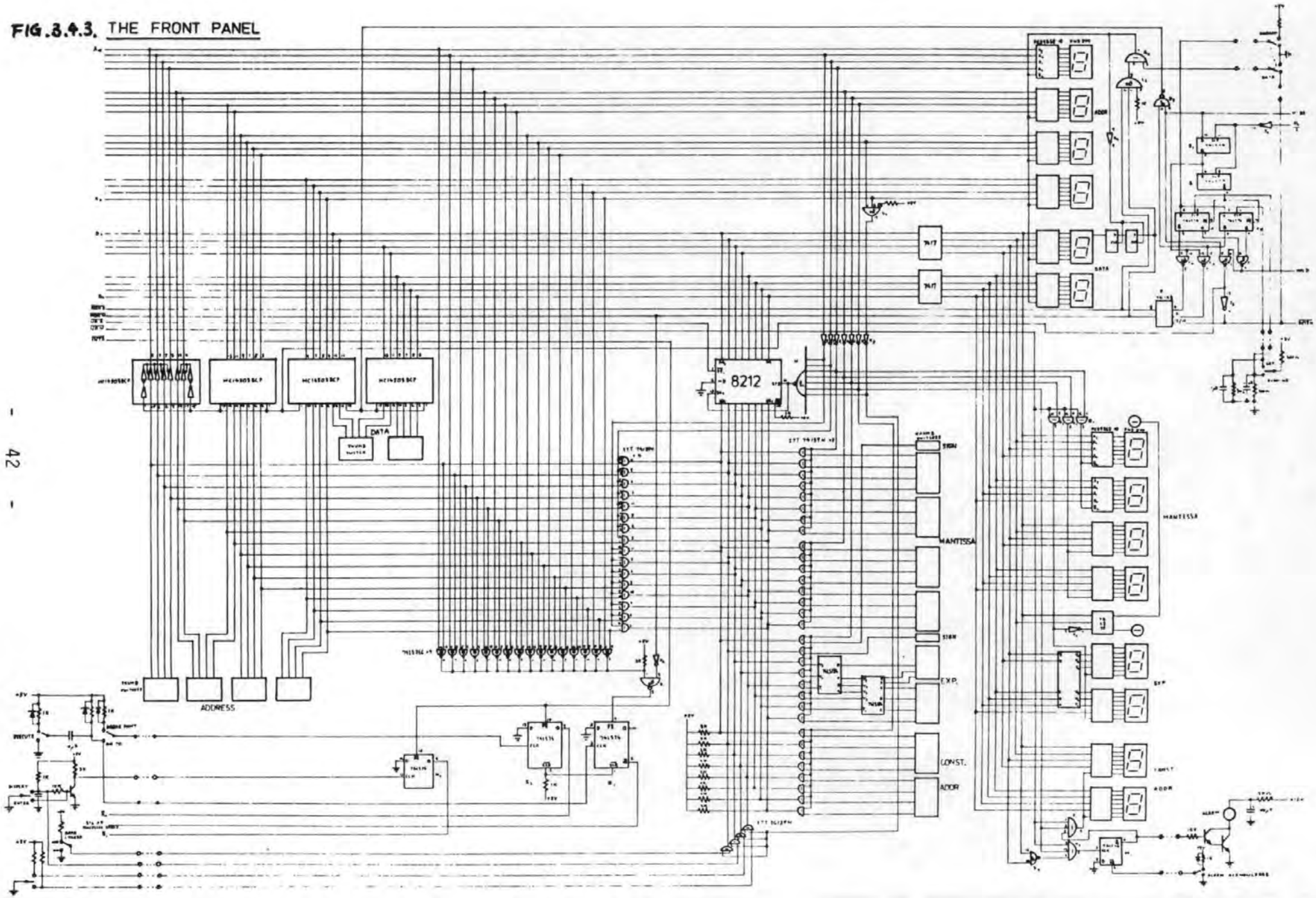


FIG.3.4.4. THE PROCESSOR BOARD

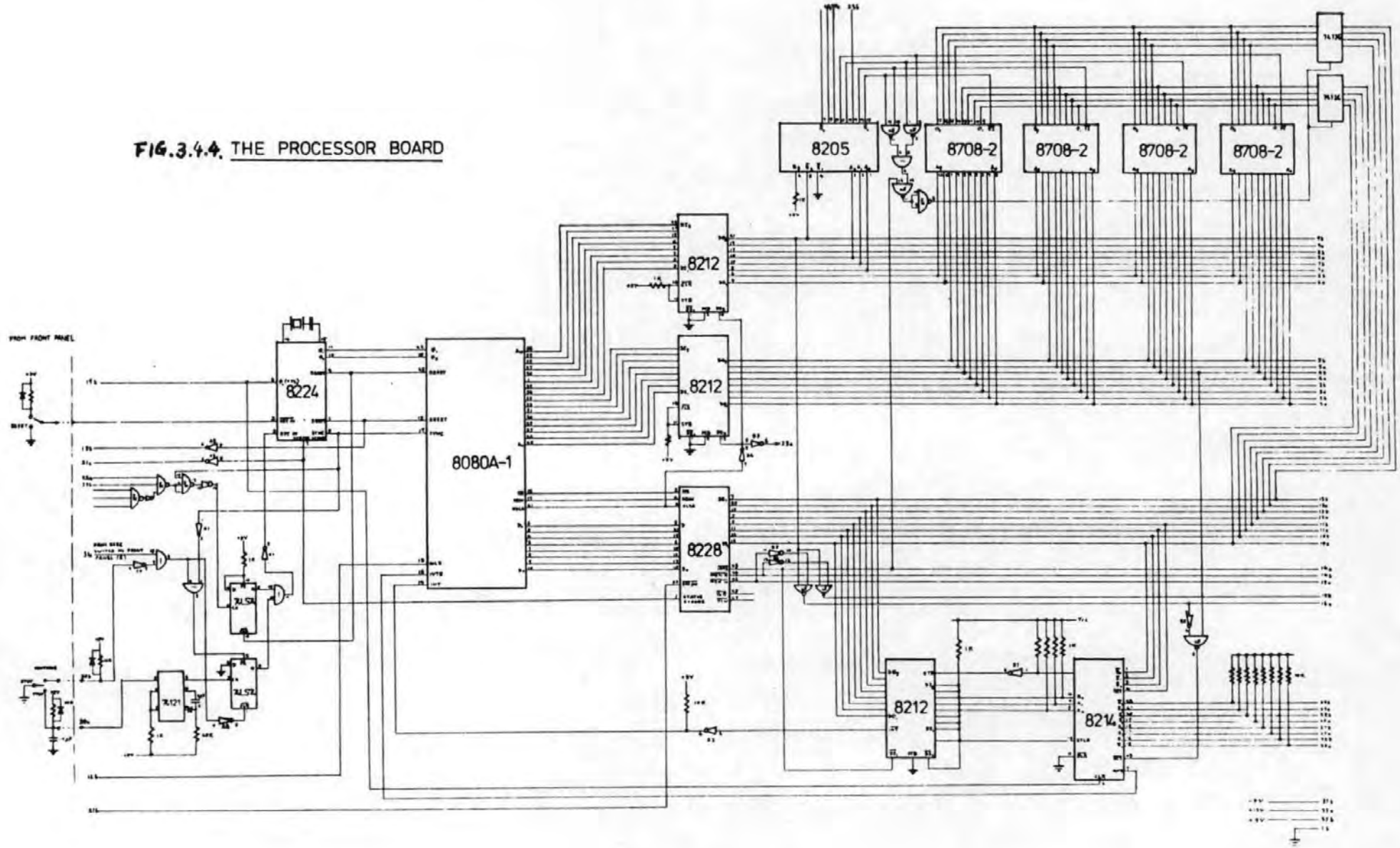


FIG. 3.4,5.

THE I/O INTERFACE BOARD

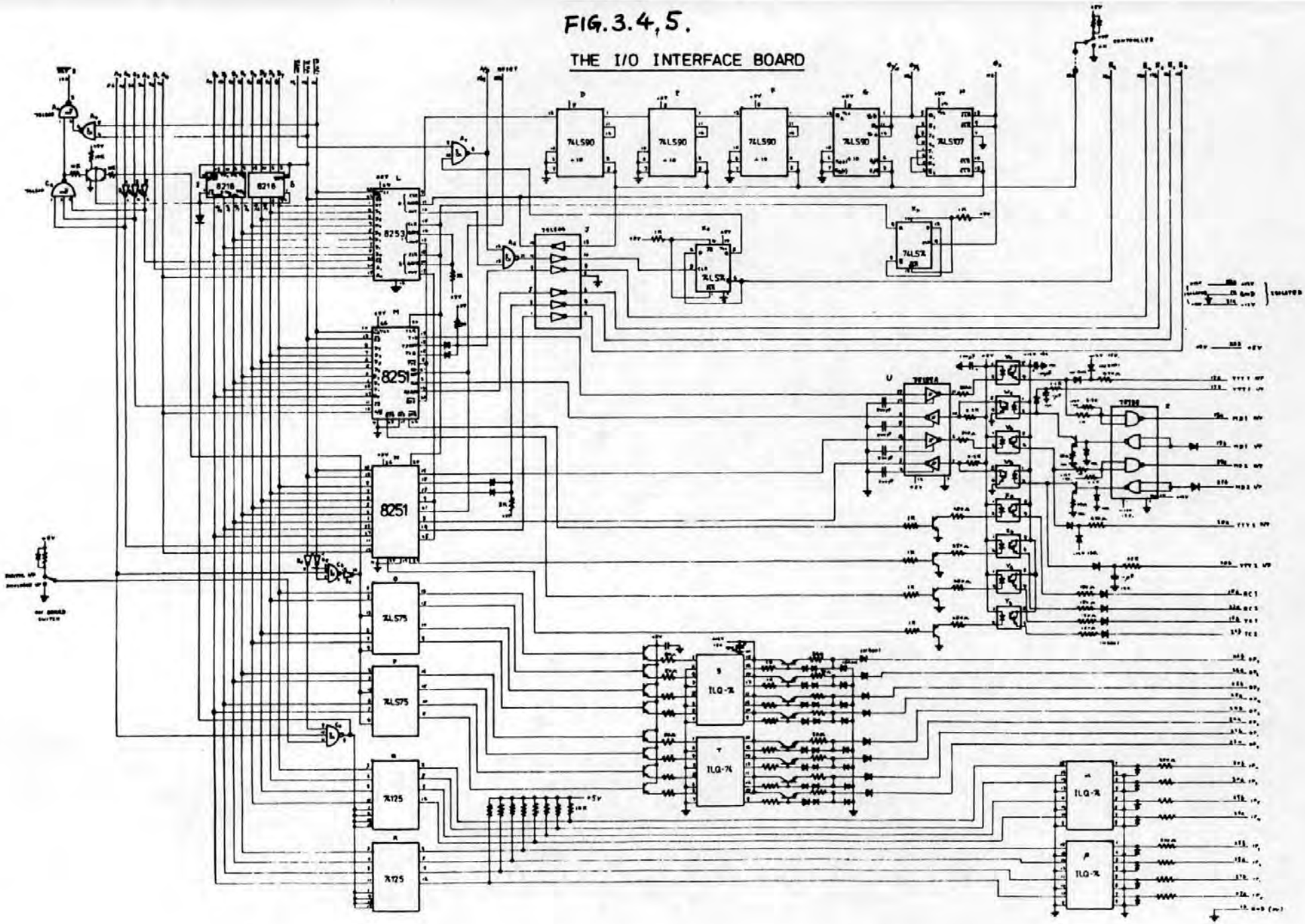
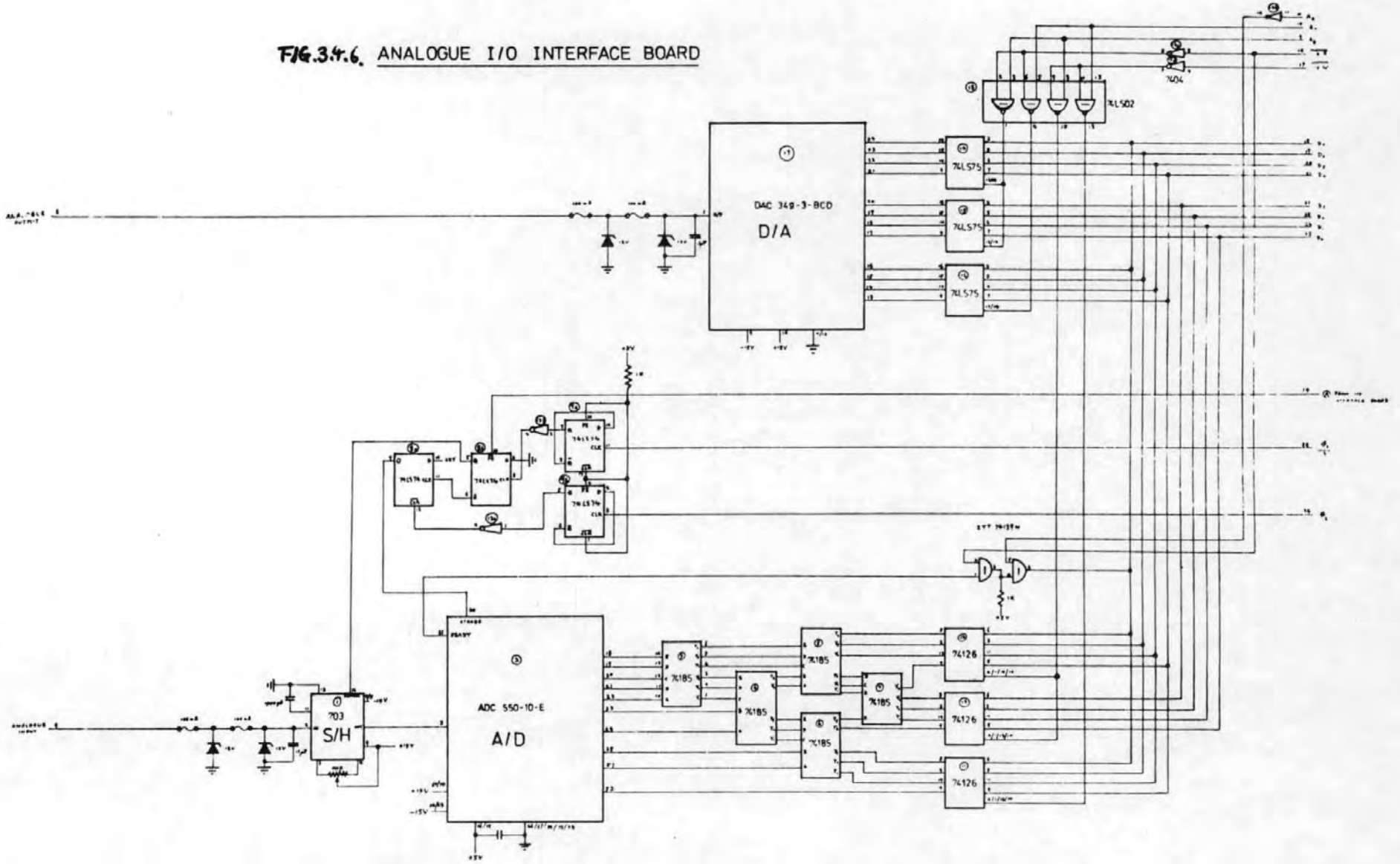
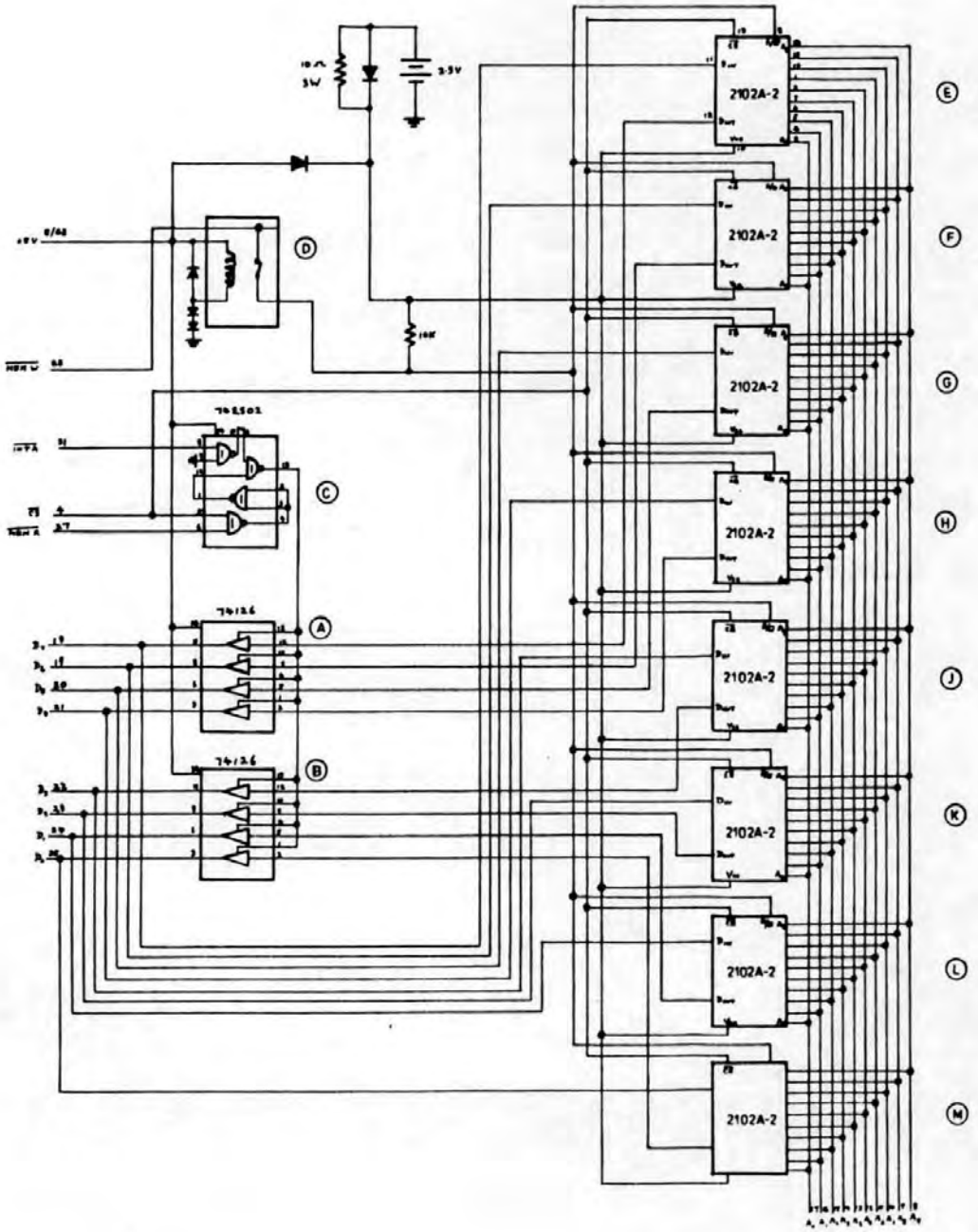


FIG.3.4.6. ANALOGUE I/O INTERFACE BOARD





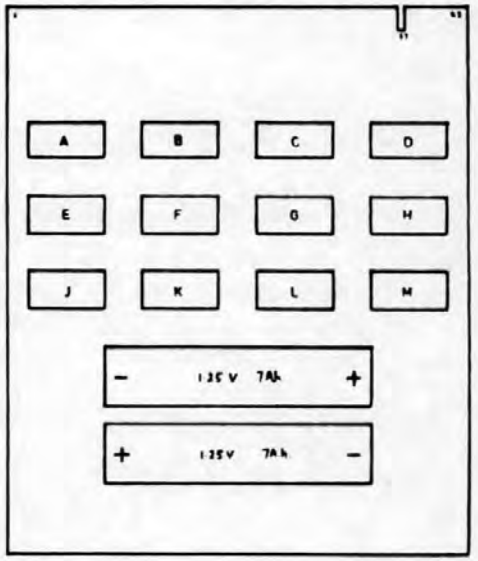


FIG. 24.7 THE R.A.M. BOARD

4. CONTROL ALGORITHM ANALYSIS AND DESIGN

In chapter two models of the plant derived by various approaches were established. From these models, control algorithms were developed. We shall examine control algorithms previously developed for the plant and some recently developed algorithms for the plant base on new understanding of the plant dynamics.

4.1 The Minimal Prototype Control Algorithm

A minimal prototype control algorithm was previously developed for the plant. It was derived base on the result of earlier modelling approach as described in section 2.2.1. From the model (2.2.9) a controller with smooth response can be derived using the method of pole zero manipulation.

Consider the overall transfer function

$$H(Z) = \frac{DG}{1 + DG} \tag{4.1.1}$$

from which we get the design formula

$$D(Z) = \frac{1}{G(Z)} \frac{H(Z)}{1 - H(Z)} \tag{4.1.2}$$

The idea was to produce a D(Z) which will cancel the plant effect and add whatever is necessary to give the desired response.

The causality and stability requirement on H(Z)

gives the desired response of the process plus controller as follows:-

$$H(Z) = \frac{Z^{-1}(1 + Z^{-1} + Z^{-2} + \ldots\ldots + Z^{-(m-1)})\ (p + q\ Z^{-1})}{m(p + q)} \qquad (4.1.3)$$

This desired response is achieved by setting Z = 1 in particular terms to remove ringing poles. The resultant controller to give the above close loop response is as follows:-

$$D(Z) = \frac{\frac{1}{Km(p + q)}\ (1 - e^{-\frac{T}{\theta}} Z^{-1})}{1 - \frac{p}{m(p + q)} Z^{-1} - \frac{1}{m} Z^{-2} - \frac{1}{m} Z^{-3} - \ldots\ldots - \frac{q}{m(p + q)} Z^{-(m+1)}} \qquad (4.1.4)$$

Although the controller in equation 4.1.4 gives smooth response, when saturated due to occasional overshoot it will be confused. An algorithm was written to take care of this problem simply by replacing all the data with a maximum value when the output was saturated.

4.2 Minimal Prototype Controller Analysis

In section 4.1 the minimal prototype controller was derived base on the previously derived plant model. However, the plant model derivation was questionable as explained in chapter 2. Nevertheless, as shown in later chapters, the performance of the above controller is no worse then the original analogue controller. The reason for this controversy can be explained as follows:-

Consider the controller in equation 4.1.4. If we choose a sampling rate such that m = 5 the controller output becomes:-

$$C = a_o E + a_1 E Z^{-1} + b_1 C Z^{-1} + b_2 C Z^{-2} + b_3 C Z^{-3} + b_4 C Z^{-4} + b_4 C Z^{-4} + b_5 C Z^{-5} + b_6 C Z^{-6} \qquad (4.2.1)$$

Where C = Controller output

E = controller input (error signal)

$$a_o = \frac{1}{K\, m(p + q)}$$

$$a_1 = -\frac{e^{-\frac{T}{\theta}}}{K\, m(p + q)}$$

$$b_1 = \frac{p}{m(p + q)}$$

$$b_2 = b_3 = b_4 = b_5 = \frac{1}{m}$$

$$b_6 = \frac{q}{m(p + q)}$$

Now consider a simple incremental digital proportional and integral (P & I) controller.

$$C = K_p\left(\frac{1}{\tau_I} \int E\, dt + E\right)$$

$$\frac{dC}{dt} = K_p\left(\frac{E}{\tau_I} + \frac{dE}{dt}\right)$$

$$\Delta C = \frac{dC}{dt} T = K_p\left(\frac{T}{\tau_I} E + T \frac{dE}{dt}\right)$$

Where T = sampling period

C = controller output

E = controller input (error signal)

K_p and τ_I are constants.

Using simple backward difference technique:-

$$\frac{dE}{dt} = \frac{E(n) - E(n-1)}{T} = \frac{\Delta E}{T}$$

$$\therefore \Delta C = K_p\left(\frac{T}{\tau_I} E + T \frac{\Delta E}{T}\right)$$

$$= K_I\, T\, E(n) + K_p(E(n) - E(n-1))$$

$$= (K_I\, T + K_p)\, E(n) - K_p\, E(n-1)$$

Now $C = C\, Z^{-1} + \Delta C$

$$= C\, Z^{-1} + K_1\, E + K_2\, E\, Z^{-1} \qquad (4.2.2)$$

Where $K_1 = K_I\, T + K_p$

$$K_2 = -\, K_p$$

Consider the general 6th order low pass digital filter. It is of the form:-

$$\hat{y} = F_1\, y + F_2\, y\, Z^{-1} + F_3\, y\, Z^{-2} + F_4\, y\, Z^{-3} + F_5\, y\, Z^{-4} + F_6\, y\, Z^{-5} \qquad (4.2.3)$$

While F_1 ..... F_6 are constants

and $F_1 + F_2 + F_3 + F_4 + F_5 + F_6 = 1$

Combining equation 4.2.2 with equation 4.2.3 and compare it with 4.2.1 we can see the similarities between them with a_o equivalent to K_1, a_1 equivalent to K_2 and the property that $b_1 + b_2 + b_3 + b_4 + b_5 + b_6 = F_1 + F_2 + F_3 + F_4 + F_5 + F_6 = 1$

From these findings, it can be intuitively concluded that the minimal prototype controlller is in the form of a digital incremental P & I controller having its last output filtered by a high order filter.

Since the original analogue controller is a P & I controller, this explains why their performances are similar in spite of the error in modelling.

4.3 Non-Linear Approach to Process Realisation and Control

Since the previous plant model was derived, there have been changes in the instrumentation of the plug level sensor. Instead of a continuous photo-resistive device giving a continuous output, the sensor was changed to one with four discrete infra-red transmitter (L.E.D.) and detectors (photo-sensitive diodes). The output of the level sensor now becomes a five steps discrete output with the required plug level at the central step as shown in Fig. 4.1.

This change of instrumentation is a very interesting development which is typical of many low cost processes in their effort to reduce cost. The object of our research have been with this type of applications in mind. Hence, it is worth studying the changes with some detail and to look at controller design from another angle - the non-linear approach.

The open loop control system can now be represented by the schematic diagram as shown in Fig. 4.2. For easier understanding of the system it would be better to describe the output of the system on the phase plane as in Fig. 4.3. The phase plane trajectories described in this chapter were being plotted by substituting values into the plant model in 2.2.23 plus a discrete time proportional and integral controller (where, $Y = L(k)$ - set point, and, $\dot{Y} = x_2 - x_1$) and in conjunction with Ref. (41) (42) (43). The major difference of this system to those mentioned in

the references is that the non-linearity here is at the system output which cannot be shifted by any form of feedback. Fig.4.4 describes an ideal case for the trajectory of the plug level when disturbed or during a start up procedure. When the plug level is at zone 'a' or 'e' the only proper action is to force it to come back into the centre as quickly as possible. After entering zone 'd' or 'b' the plug level is very near to the set point and should be decelerated in order that the plug level can settle in zone 'c'. Once the plug level entered zone 'c' its deceleration should die with the velocity to allow for the plug level to settle.

Consider the system model as shown in Fig. 4.2. Experiment with the system shown that B_R and τ would change due to fluctuation of input steam temperature and pressure, environmental changes such as ambient temperature and the changes in the properties of yarn. These lead to a stochastic system with time variant gain and dead time while ζ and ψ are the disturbances and drift respectively. With the non-linearity added at the sensor, the system become a very complex one.

4.4 Non-Linear Adaptive Controller Design.

Based on the understanding of the system as described in section 4.3, a simple controller design technique was used to study the system response on the phase plane and to design the controller accordingly.

From Fig. 4.3 first consider the control action when the plug level is in zone 'e' and 'a'. In this circumstances, we have no information as to how far away the plug level is from the set point,

the only appropriate control action is to drive the process as hard as possible to force the plug level back within the range of the sensors. Hence, under this condition, the controller works in the form of a 'bang-bang' control. In order to calculate the optimal control action it is desirable to know the velocity of the plug level when it moves back within range of the sensors (i.e. zone 'd' and 'b'). Since the controller/sensors have no knowledge of the plug position the only indication of the plug level velocity is to record the time it took to come back. Hence, some form of integral action is required in the controller for this purpose.

Secondly consider the situation when the plug level is within the range of the sensors. Imagine the control action of a digital incremental controller with only proportional gain, the system response is shown in Fig. 4.4. That is, if the proportional gain constant has been set high a large displacement from the set point would give an almost perfect response characteristic (trajectory 'f'). However, for a small displacement from the set point the plug level would oscillate between zone 'e' and 'd' as shown by trajectory 'g' and never settle at the set point. In order to avoid the unstable case as shown by trajectory 'g' we have to set the proportional gain such that the trajectory in zone 'd' is almost horizontal but never dips. While, the trajectory in zone 'c' should never rises.

Fig. 4.5 showed the trajectories of compromise setting on the controller to give fast response and no overshoot for small displacement in zone 'e' as shown by trajectory 'j' while, larger displacement in zone 'e' have minimal overshoot as shown in trajectory 'h'.

Having considered the control action when the plug level started from zone 'e' and 'a' we shall consider the situation where the plug level started from zone 'd' and 'b'. Fig. 4.6 shows the responses of the plug level with a controller having a fixed proportional gain. A large displacement from the set point within zone 'd' and 'b' would converge to the set point as shown by trajectory 'k' while smaller displacements from the set point would oscillate at the edge of the set point as shown by trajectory '1'.

To conclude the above phase plane study of the system response to different control actions,the controller porportional gain has to be different for different displacements within a zone. However, the difficulty is that the controller has no way of knowing where the plug level is within a zone. The only useful information that the controller can go by is the time between transition from one zone to another. This indicates that the controller should change its proportional gain with time and with the zone that the plug level is in.

4.4.1 Basic structure of controller.

Based on the above findings, a framework for controller can be built as shown in Fig. 4.8.

Define: The output of controller

$$c(n) = C(n-1) + \Delta C$$

and the output of the sensor

$$E(n) = q\{-2, -1, 0, 1, 2\}$$

where $q \in$ the closed and bounded set of integers

The controller consisted of a 'bang-bang' control which is fed by an averaging filter which filters out most of the noise. The 'bang-bang' controller only operates when the plug level is outside the range of

the sensors. When the plug level is within the range of the sensors, the proportional and integral control come into action. The integral controller is fed by a non-linear delay in order to minimise oscillation due to the time delay within the system. The non-linear delay only operates on the integrator because if added onto the proportional control as well, it would reduce the system close loop frequency response. A digital filter is added to the controller input to filter out noise as well as smoothing of the sensor outputs. The filter used is:

$$\hat{E}(n) = \alpha E(n) + (1 - \alpha) \hat{E}(n - 1)$$

where α = filter response constant.

4.4.2 Proportional gain tuning algorithm

Since there are only four steps through the sensors, we define max proportional gain $b_p = \frac{C \text{ max}}{4}$ where C max is equal to maximum heater output power.

When the plug level is outside the sensor range, (that is $E(n) = \pm 2$, where $E(n)$ is the displacement from the set point) heater output would go to C max or zero in order to bring the plug level back within the range of the sensors as quickly as possible. This situation is true at the initial start up of the system in order to make the system settle at the set point in the shortest time. Hence the gain settings for the system is at its maximum. After the set point is reached, we want the minimum control action in order to obtain a constant amount of crimp. Hence, the controller at steady state has to be detuned to minimise its control action. Hence, the algorithm for tuning of the integral and proportional gain (K_I and K_p) are as

follows:-

(1) If $|E(n)| = 2$, $K_I = b_I$, $K_p = b_p$.

(2) If $|E(n)| = 1$, no change to K_I and K_p.

(3) If $|E(n)| = 0$, i.e. at set point.

(a) If $b_I > K_I > a_I$, then $K_I(n) = K_I(n-1) - \frac{b_I - a_I}{4T_s} T$

If $K_I < a_I$, then $K_I = a_I$

Where $b_I = \frac{C \text{ max}}{8 \text{ Tm}}$ = max integral gain,

$a_I = \frac{C \text{ max}}{16 \text{ Tm}}$ = min integral gain.

Tm = period of oscillation of output

T_s = nominal time constant for start up

(b) If $b_p > K_p > a_p$, then $K_p(n) = K_p(n-1) - \frac{b_p - a_p}{4T_s} T$

If $K_p \leq a_p$, then $K_p = a_p$

Where $b_p = \frac{C \text{ max}}{4}$, $a_p = \frac{b_p}{6}$

Real-time application of this control algorithm can be found in chapter 5. A schematic diagram of the complete non-linear adaptive controller is as shown in Fig. 4.9.

Fig. 4.1 Characteristic of the Plug Level Sensor

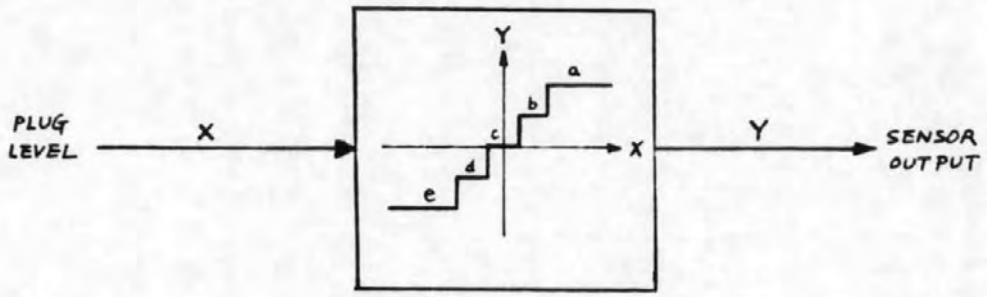


Fig. 4.2 Block Diagram of The Process

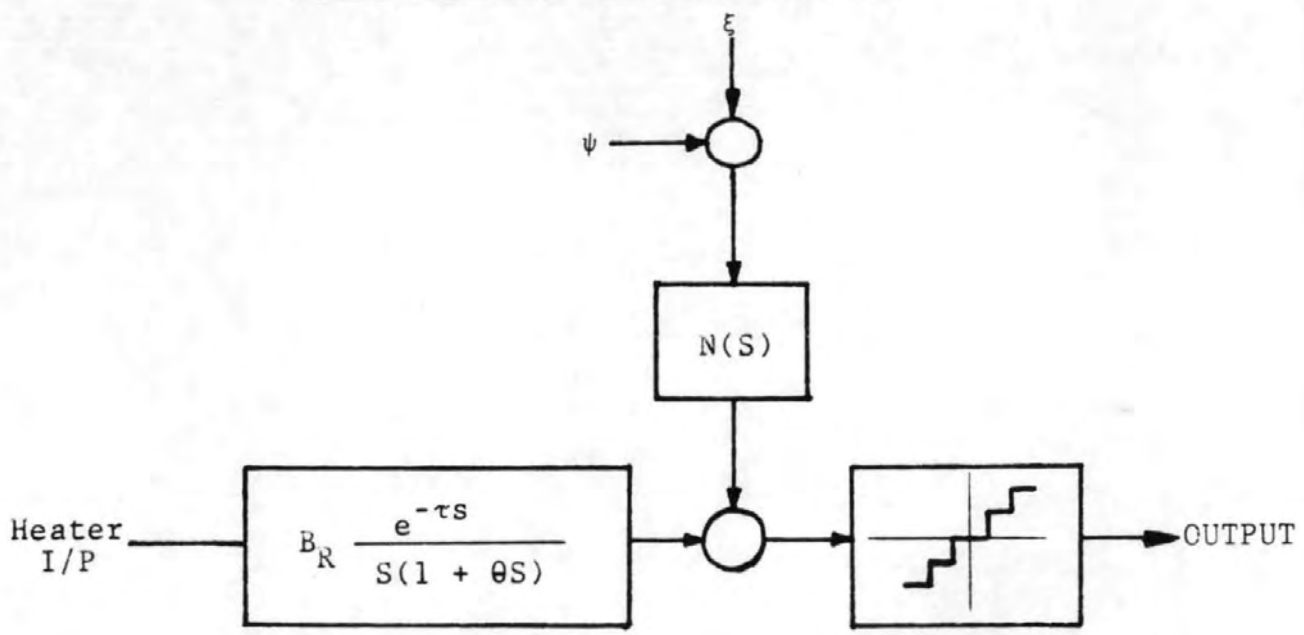
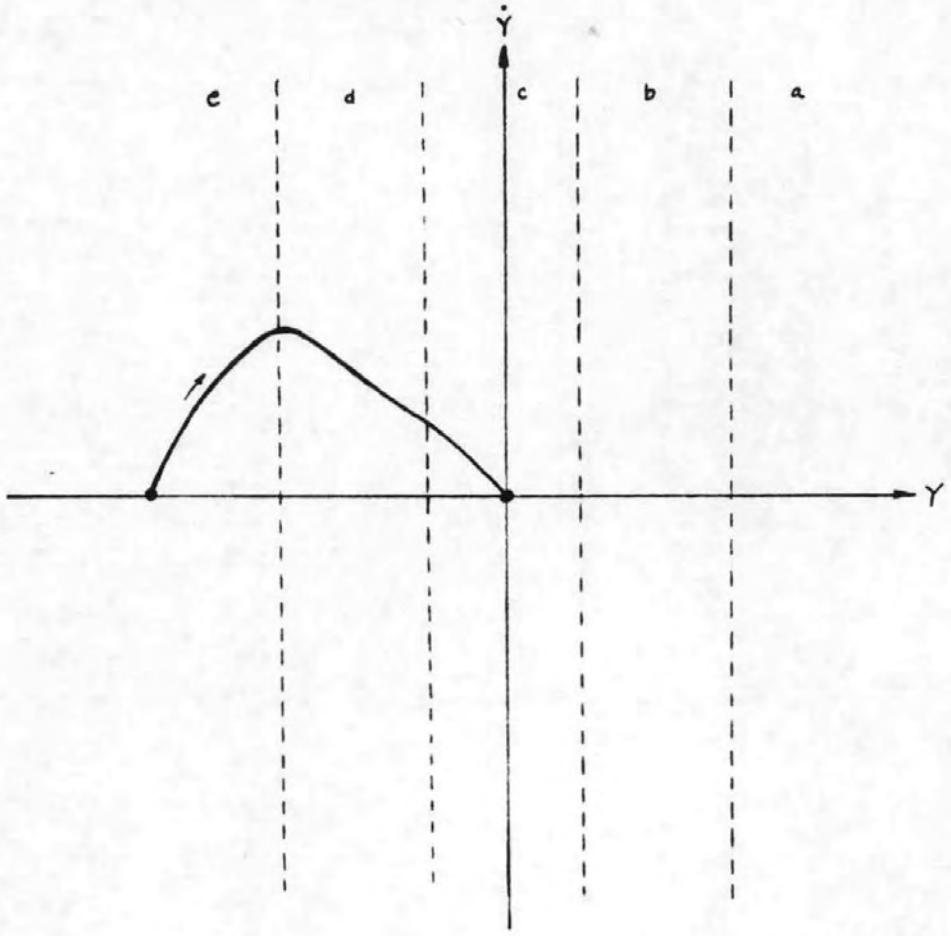


Fig. 4.3 Ideal Trajectory of the Plug Level



N.B. Y= displacement of plug level from set point.

Fig. 4.4 Trajectory of the Plug Level with Proportional Gain of Digital Incremental Controller too High

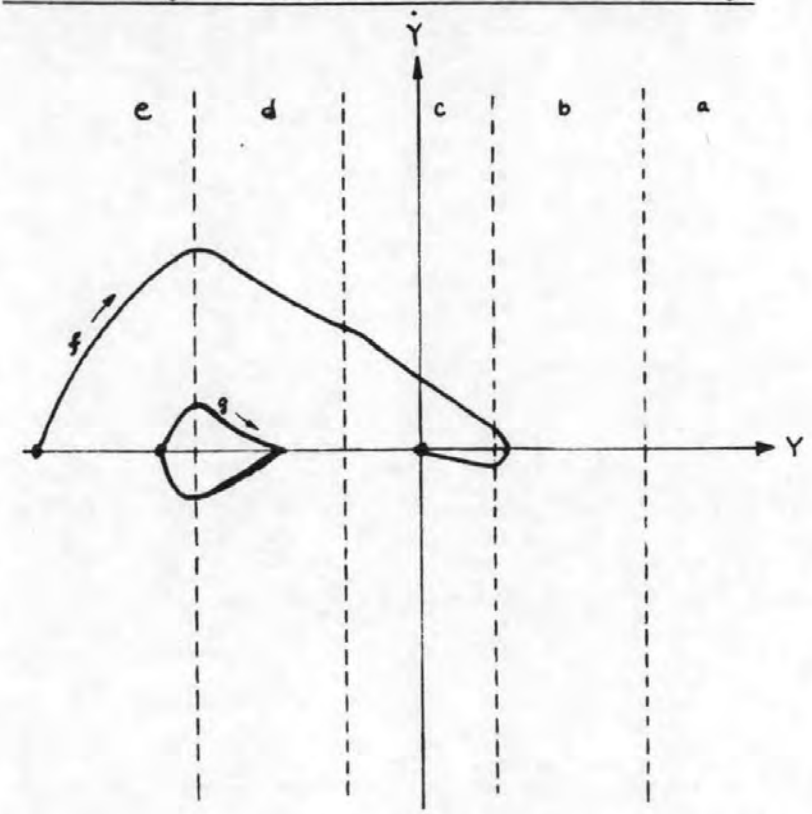


Fig. 4.5 Trajectory of the Plug Level with a Well Tuned Controller for Level at Zone 'e' and 'a'

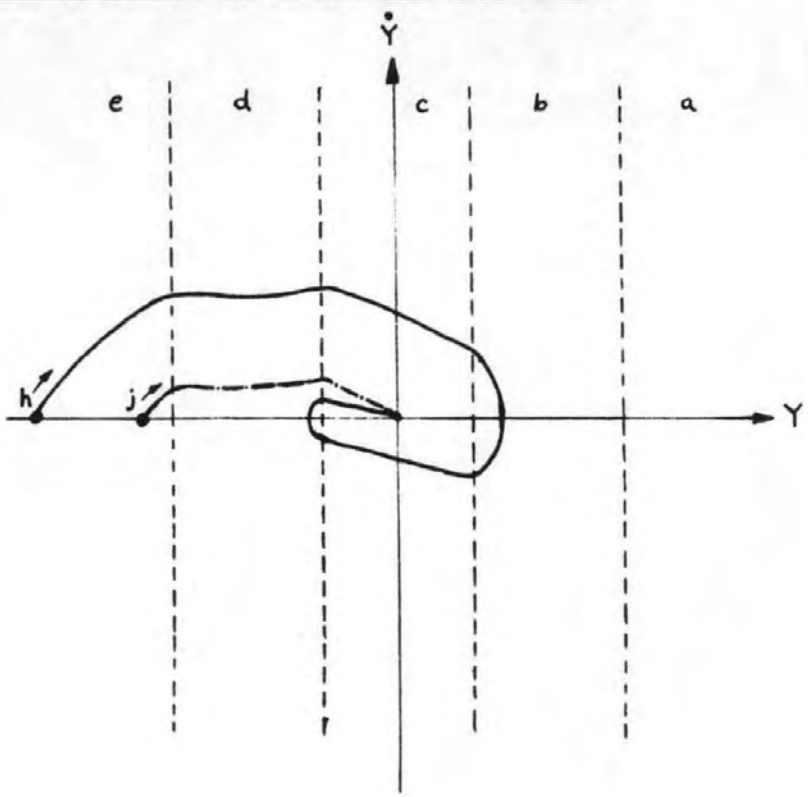


Fig. 4.6 Trajectory of Plug Level from Zone 'd' and 'b' (i.e. small displacement)

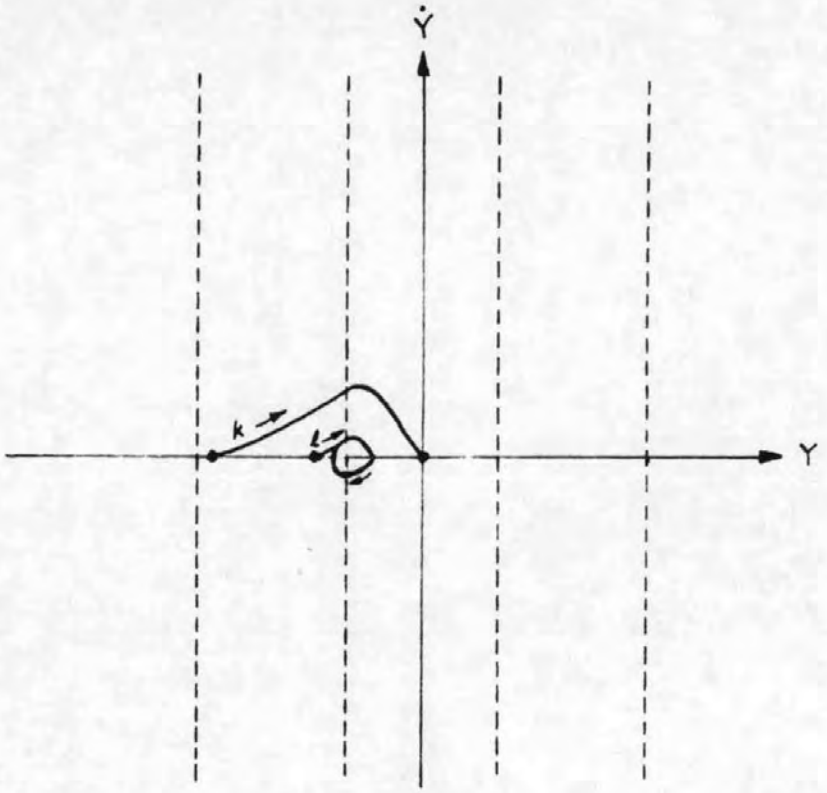


Fig. 4.7 Trajectory of Plug Level with Changing Gain and Integral Rate

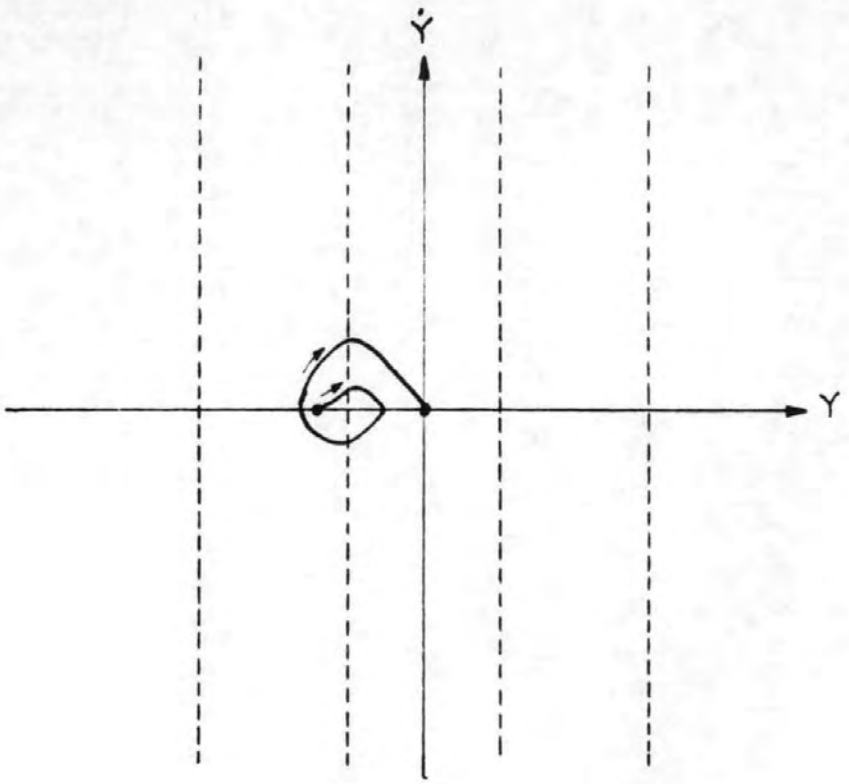


Fig. 4.8 Schematic Diagram for Basic Frame-work of Non-linear Adaptive Controller

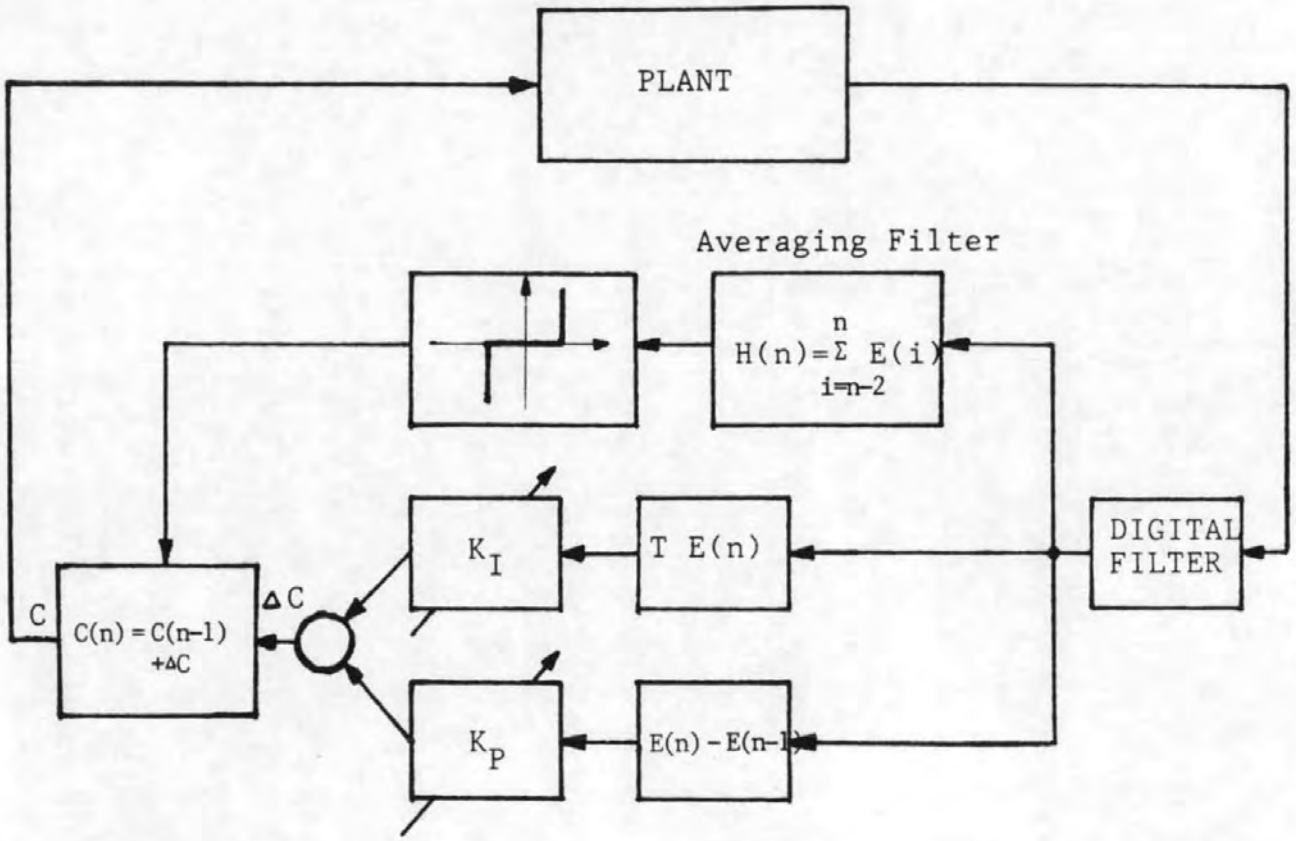
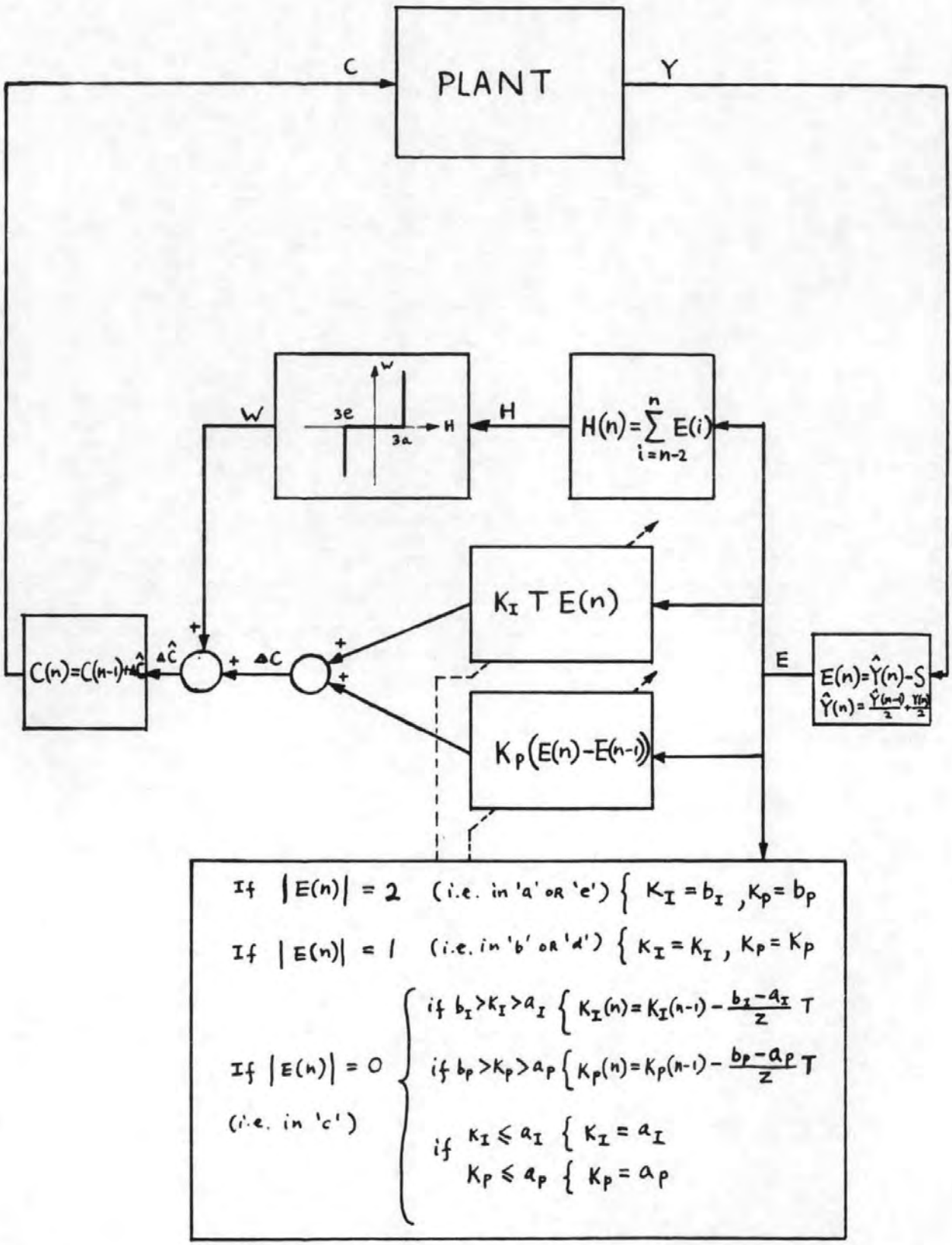


Fig. 4.9 Schematic Diagram of Non-linear Adaptive Controller.



5. INDENTIFICATION AND ON-LINE IMPLEMENTATION OF CONTROL ALGORITHMS ON THE ACTUAL PROCESS

5.1 Introduction

In this chapter efforts made to find out how close the model derived in chapter 2 resembles the plant operations are presented. Although there are various advanced identififcation algorithms available they are not always applicable in some plant. The difficulty in applying identification methods to the plant under study here was that, its output was only in five discrete steps while its input was in sixty four discrete steps.

In the second half of this chapter comparison of on-line performance, of the control algorithm derived in chapter 4 to that of an analogue P & I controller is presented. It also includes interface design and software structure for the implementation of the control algorithm.

5.2 System Identification

In this section, we describe the effort made to relate the system models derived to the real plant. Although the recent advances in identification techniques provided a number of modern identification methods (Ref. 29) and (Ref. 30), their end products are normally in the form of differential equations which if the system is non-linear or has a dead time may not be easily converted to the transfer function

form for comparision with the derived model. Hence, "classical" identification methods were used to identify the plant (Ref. 47). As explained in chapter 2, the output level of the plant is measured in five discrete steps. Thus identification methods based on the sinusoid response of the plant are not suitable. Attempts have been made to install a linear transducer to measure the plug level, but this was unsucessful due to the working environment. In view of the above problems, the more sensible approach was to inject white noise into the system. By cross-correlating the input with output, the plant impulse response is obtained. Although the disturbance within the system and, the measurement and quantisation noise at the output, affect the result, the variance of these noises are much less than the test signal.

In order to inject white noise into the system, several ways of generating pseudo-random binary sequences were examined including methods described in (Ref. 31), (Ref. 32) and a modified version of an algorithm described in (Ref. 33). Ultimately, it was decided to use the simple method of 'exclusive OR' feedback shift register chain for its simplicity, zero mean and whiteness. In this case, a chain of nine registers was used. The schematic representation of the generator is shown in Fig. 5.1. The actual program for the generator and the program for testing maximum cycle length can be found in Appendix 8.

Finally, the output of the generator was fed into a correlator to obtain its autocorrelation function. The output of the correlator can be found in Fig. 5.2.

The pseudo-random binary generator program was then put into the "Algorithm Development/Implementation System" and used to inject noise into the input(heater) of the plant. The plant was first brought to steady state by the 3-term controller inside the "Algorithm Development/ Implement System". When the steady state of the plant had been reached for some time, the control was then switched to the pseudo-random binary sequence generator program and the input and output of the plant were logged by the data logger within the microprocessor system and punched on paper tape. A sampling period of 4 seconds was used for both control and data logging. The paper tape was brought back to the college and analysed using a cross-correlation and auto-correlation program written for the microprocessor system as shown in Appendix 9. The cross-correlation function of the system was output both in Hexdecimal numbers and in graph form. The output of the correlation program can be found in Fig. 5.3. Note that the cross-correlation graphs are all inverted. One important consideration for this experiment is to choose the right amplitude of input signal and the right sampling rate Fig. 5.3 (a) showed the cross-correlation of one set of experimental results. The output is virtually indistinguishable from noise because the input signal is too small. With the non-linearity at the output, the quantisation noise is very significant when the input signal is low. However, if the input signal is too high, the output saturates, hence, inducing large errors in the identification. Fig. 5.3.(c) and (e) are the result of well chosen input signal levels. Although the noise level is still quite high we can extract the required information. The sampling period is also important. High correlation between samples was evident if the sampling period is too small. In our case, the sampling period was 4 seconds,

a compromise between the response of the system and the correlation between samples.

During the course of the experiment the system would tend to drift. Hence, the mean of the output would drift which affects the result of the identification. This difficulty was overcome by putting a high pass filter before the cross-correlation program eliminating any drift. The algorithm for the high pass filter is of the form:

$$Y_{k+1} = U_{k+1} - \frac{1}{T} \sum_{n=0}^{k} Y_n .$$

Where: Y = filter output

U = filter input

T = time constant of the filter

Fig. 5.3 (c) and (e) are the result of the same set of data after passing through H.P. filter of different time constant. Fig. 5.3 (e) has a well tuned H.P. filter with the time constant of 160 sec., while, Fig. 5.3 (c) has a time constant of 320 sec. for the high pass filter. We can see in Fig. 5.3 (c) and the auto-correlation of the same input signal in Fig. 5.3 (d) that drift of the O/P signal affected the result. This effect was eliminated in Fig. 5.3 (e). The impulse response of the nylon crimping plant obtained from the above identification method is as shown in Fig. 5.3 (g) which is derived from Fig. 5.3 (e).

Now consider the transfer function in (2.2.8) - ignoring the noise term.

$$L(s) = \frac{-K_1}{1 + s\theta} \left(\frac{1 - e^{-ts}}{s}\right) U(s)$$

The impulse response of the above equation can be obtained by taking the inverse Laplace transform of the transfer function as follows:

$$L(s) = -\frac{K_1}{(1 + s\,\theta)} + \frac{K_1\, e^{-\tau s}}{(1 + s\,\theta)}$$

$$\mathcal{L}^{-1}(L(t)) = -\frac{K_1}{\theta}\int_0^t e^{\frac{-t}{\theta}}\, d + \frac{K_1}{\theta}\left[\int_0^{t-\tau} e^{\frac{-(t-\tau)}{\theta}}\, dT\right] H(t - \tau)$$

$$= K_1\left[\left(1 - e^{\frac{-t}{\theta}}\right) - \left(1 - e^{\frac{-(t-\tau)}{\theta}}\right) H(t - \tau)\right] \qquad (5.2.1)$$

This equation was written in BASIC and fed into the PDP11 to obtain the impulse response of the model with various setting of τ and θ output in graph form as shown in Fig. 5.4(a).

Now consider the model derived in section 2.2.2. From equation (2.2.19), the variation in Q_b is so small in comparison to the variation in $\dot{m}_o$ that it can be ignored. This give a model approximating the process as follows:

$$L(s) = B_R \frac{e^{-\tau s}}{s(1 + \theta s)} E_i \qquad (5.2.2)$$

Where B_R = Constant.

The impulse response of the above equation is as shown in 5.4 (b). Comparing it with Fig. 5.3 (g) the value of τ and θ are obtained as 10 sec. and 32 sec. respectively. Comparing the impulse responses, the response of (5.2.2) looks much nearer to that obtained from the real plant. And the drift associated with the output is typical of (5.2.2) as explained in section 2.2.

The above interpretation of the correlation output is mainly due to the prior knowledge of the impulse response. In real life,

the dynamics of the system are limited by the physical construction of the plant and should behave more or less as expected. Hence, the above interpretation is not without

Furthermore, consider the derivation of the relationship between the cross-correlation and the impulse response:
Assume the following discrete-time system to be completely controllable and observable.

$$x(k+1) = F\,x(k) + b\,u(k) + g\,w(k),\quad F = \left[\begin{array}{c|c} 0 & I \\ \hline f^T & \end{array}\right],\quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix},\quad g = \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}$$

$$z(k) = h^T x(k) + v(k)$$

$$h = \begin{bmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \qquad (5.2.3)$$

Where: $x(k)$ = n - dimensional vector

$u(k)$ and $z(k)$ are the scalar input and output respectively.

$w(k)$ and $v(k)$ are independent and identically distributed noise sequences.

F, b, g, and h are unknown matrices of appropriate dimensions. The transfer function representation of the system for zero-initial condition and b = g is:

$$z(k) = \sum_{j=0}^{k-1} h^T F^j b\,[u(k-j-1) + w(k-j-1)] + v(k)$$

$$= \sum_{j=0}^{k-1} c(j)\,[u(k-j-1) + w(k-j-1)] + v(k) \qquad (5.2.4)$$

Where $c(j) = h^T F^j b$ is the impulse response of the system.

If a discrete interval binary noise is applied at the input u(k) and output z(k) is:

$$R_{zu}(m) = E\{z(N)\, u(N - m)\}$$

$$= E\{ \sum_{j=0}^{N-1} C(j)[u(N - j - 1) + w(N - j - 1)]\, u(N - m)\}$$

$$= \sum_{j=0}^{N-1} C(j)\, E\{u(N- j - 1)\, u(N - m)\}$$

$$\triangleq \sum_{j=0}^{N-1} C(j)\, R_{uu}(m - j - 1) \qquad (5.2.5)$$

This proved that if the auto-correlation of the injected noise is an impulse, the cross-correlation of input and output ($R_{zu}(m)$) is the impulse response of the system.

Now by stacking equations (5.2.3) for m = 1, 2, ..., N

$$F_{zu}(N) = \begin{bmatrix} R_{zu}(1) \\ \vdots \\ R_{zu}(N) \end{bmatrix} = \begin{bmatrix} R_{uu}(0) & \cdots & R_{uu}(-N+1) \\ \vdots & \vdots & \vdots \\ R_{uu}(n-1) & \cdots & R_{uu}(0) \end{bmatrix} \begin{bmatrix} C(0) \\ \vdots \\ C(N-1) \end{bmatrix}$$

$$= F_{uu}(N)\, C(N - 1) \qquad (5.2.6)$$

From 5.2.4 we can intuitively conclude that the cross-correlation sequence $F_{zu}(N)$ (or function) output representing the impulse response of the system is affected by the input sequence if the input sequence is not a perfect impulse. Using this result, we can compare the auto-correlation of the input noise as in Fig. 5.3 (f) with the cross-correlation output in Fig. 5.3 (e) to explain the wide variance of

the impulse response graph of the impulse in Fig. 5.3 (f). Theorectically we should be able to eliminate this effect as follows:

$$C(N - 1) = [F_{uu}(N)]^{-1} \qquad (5.2.7)$$

However, the amount of computation required in order to calculate and invert $F_{uu}(N)$ is enormous and could only be tackled by very powerful computers.

5.3 Practical Implementation of the Controller

5.3.1 The original analogue controller

The circuit diagram of the original analogue controller is shown in Fig. 5.6. The circuit consists of three operational amplifiers. One for the proportional gain, one for the integral action and a third one for summing the two. The inputs from the level sensor are weighted by a resistor chain before connecting to the operational amplifier circuit. The input from the top sensor is used to switch the heater fully on in order to force the plug of yarn down. The output of the third operational amplifier is used to set the firing rate of the triac.

5.3.2 The interface from microprocessor system to the plant

In order to implement the digital control algorithm, an interface was built to fit into the existing rack which houses the analogue controller. The circuit of this interface can be found in Fig. 5.9. It was designed to fit the existing specifications of the plant such as

input/output of the level sensor and heater. A schematic-diagram showing the function of the device can be found in Fig. 5.5. This interface has a much better noise rejection than the original analogue controller. It does not require any 100 Hz pulse input saving the cost of an accurate pulse generator. The a.c. voltage is being switched at its cross-over point minimising spikes created in the mains. The interface consists of two main parts, one for converting the level into a binary signal while the other is for converting the binary output to the corresponding heater load.

5.3.3 The three-term controller

For elementary experiments with the plant, a general purpose three-term control algorithm was written for the microprocessor system to control the plant. The program for this algorithm can be found in Appendix 10. The input of the controller from the interface is first passed through a digital recursive low pass filter of the form

$$\hat{Y}_{k+1} = (1 - K)\,\hat{Y}_k + K\,Y_{k+1}$$

Where $\hat{Y}$ = filter output

Y = filter input

K = filter constant and have a value between 1 and 0

to get rid of any noise generated after the hardwired low pass filter before the signal is fed into the three-term controller. The digital three-term controller algorithm used is:

$$C_t = C_{t-1} + \Delta C$$

$$\Delta C = K_I\,T_s\,E_t + K_p\,(E_t - E_{t-1}) + K_D\,(E_t - 2\,E_{t-1} + E_{t-2})\,T_s^{-1}$$

Where C = Controller output

$\Delta C = C_{t+1} - C_t$

K_p = Proportional gain

$K_I = \frac{K_p}{T_I}$

T_I = integral time constant

$K_D = K_p T_D$

T_D = differential time constant

T_s = sampling interval

This 3-term control program is a general purpose program. It can accept input in 8 bit binary form or in analogue form converted by the A/D converter into 3 digit B.C.D. The output of the controller can be in 8 bit binary or 3 digit B.C.D. for the D/A converter to transform it to an analogue output signal. The constants for the program are stored in the constant table in the following locations:

Address	Constant
01	Controller o/p
02	Controller i/p
03	Set point
04	K_p
05	K_I
06	K_D
07	E_t
08	E_{t-1}
09	E_{t-2}
10	K

All the constants are stored in form of a 4 digit fractional part and a 2 digit exponent except for the controller I/P and O/P. The controller O/P and I/P are stored in address '01' and '02' in the form of either a 2 digit Hexdecimal number at the least significant digits of the mantissa with the exponent set to '1' or a 3 digit B.C.D. at the most significant digits of the mantissa with the exponent set to '0'. All the above constant are addressable from the control engineers' panel on the Algorithm Development/Implementation System. Experiments with the plant using this program set to accept binary input and output showed that it works well with the plant and the control is very similar to that of the original analogue controller.

5.3.4 The non-linear adaptive controller

Further experiment with the plant showed that improvement can be made to the 3-term controller to give better control for the plant as explained in chapter 1. The Non-linear Adaptive Control Algorithm was converted into 8080 assembler code (Appendix 11) and implemented by the "Program Development/Implementation System". This program is dedicated entirely to the process under investigation. It is so designed that it would fit into the microprocessor based controller for the plant with minimum modification. It works in binary instead of floating point B.C.D. as in the case of the general purpose 3-term controller. The result obtained using this algorithm are very encouraging as can be seen in the next section.

The British Library LENDING DIVISION
Boston Spa, Wetherby, West Yorks LS23 7BQ

PLYMOUTH POLY.

PhD Thesis by YUNG, K.L.

We have given the above thesis the Lending Division identification number:

D 57058/85

In your notification to Aslib please show this number, so that it can be included in their published 'Index to Theses...' and 'Abstracts of Theses', respectively.

QAB

J P CHILLAG
Theses Officer

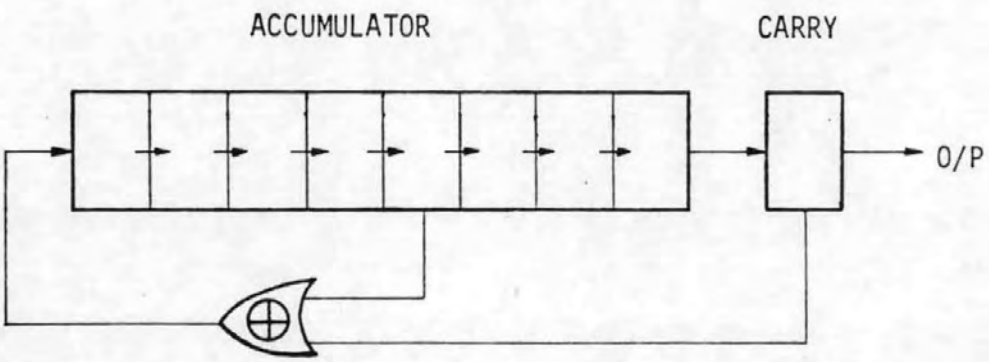
5.3.5 Comparison of the analogue controller with the non-linear adaptive control algorithm

In order to compare the control performance of the two controllers the best way is to compare the dynamics of the plug level at start up and under disturbance at steady state. However, due to the sensor problem as explained previously, there is no way of logging the position of the plug level unless it is within the range of the level sensors. Another way of finding out the ability of the controller is to look at the temperature fluctuation of the steam temperature inside the jet where a thermocouple has been installed. This method is reliable because the operators had been using it to determine the quality of crimped yarn. Typical runs controlled by the analogue controller and the non-linear adaptive controller can be found in Fig. 5.8 and Fig. 5.7 respectively. Comparing the two graphs, we can see that the non-linear adaptive controller has a much shorter start up time to achieve steady state. From this evidence, we can conclude that the non-linear adaptive controller controls the plant better than the original analogue controller.

5.4 Conclusion

In this chapter, we see the identification of the control loop and the practical implementation of the control algorithm on the nylon crimping plant. All these operations were performed on the Algorithm Development/Implementation System descripted in chapter 3. Without the flexibility it offers, all this work would be very difficult or impossible to carry out.

Fig. 5.1. 'Exculsive OR' Feedback Pseudorandom Binary Sequence Generator.

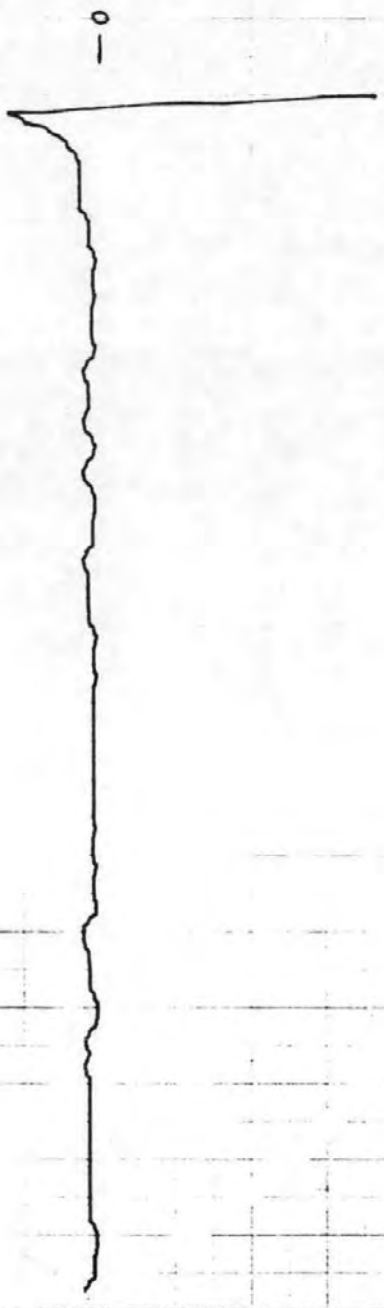


N.B. The nine registers so arranged here are for simulation on the microporcessor being eight bits in the accumulator and one bit in the "CARRY" flag.

Fig. 5.2 Autocorrelation Function of the Pseudo-random Binary Sequence.

Setting for the correlator: Input: .4 to 1 $V_{r.m.s.}$
Function: Autocorrelation
Averaging: Summation, N=2×1024 samples
Timescale: 100 ms/mmΔt

Input Signal for the P.R.B.S. was 0.2 s/sample, level between 0 and 1.3 V.



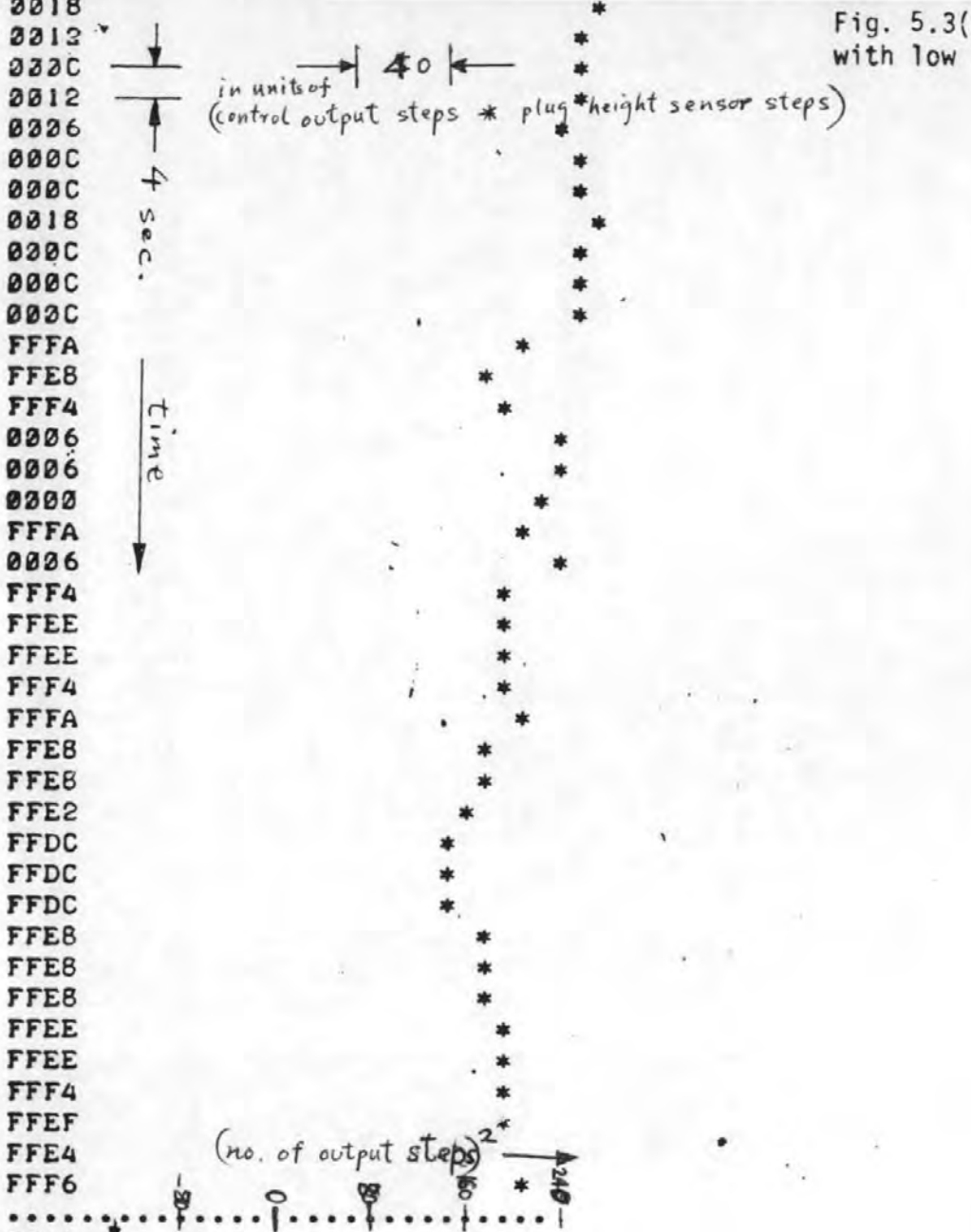


Fig. 5.3(a) Cross-correlation with low level input.

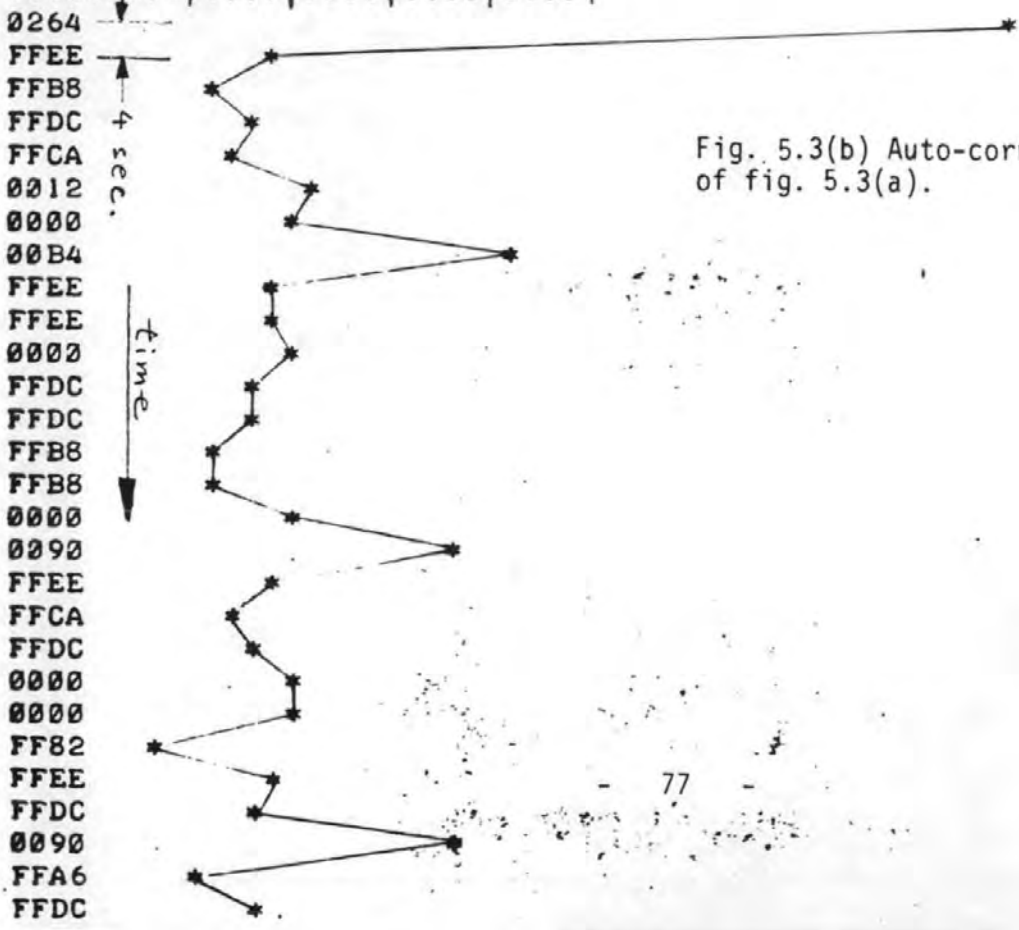
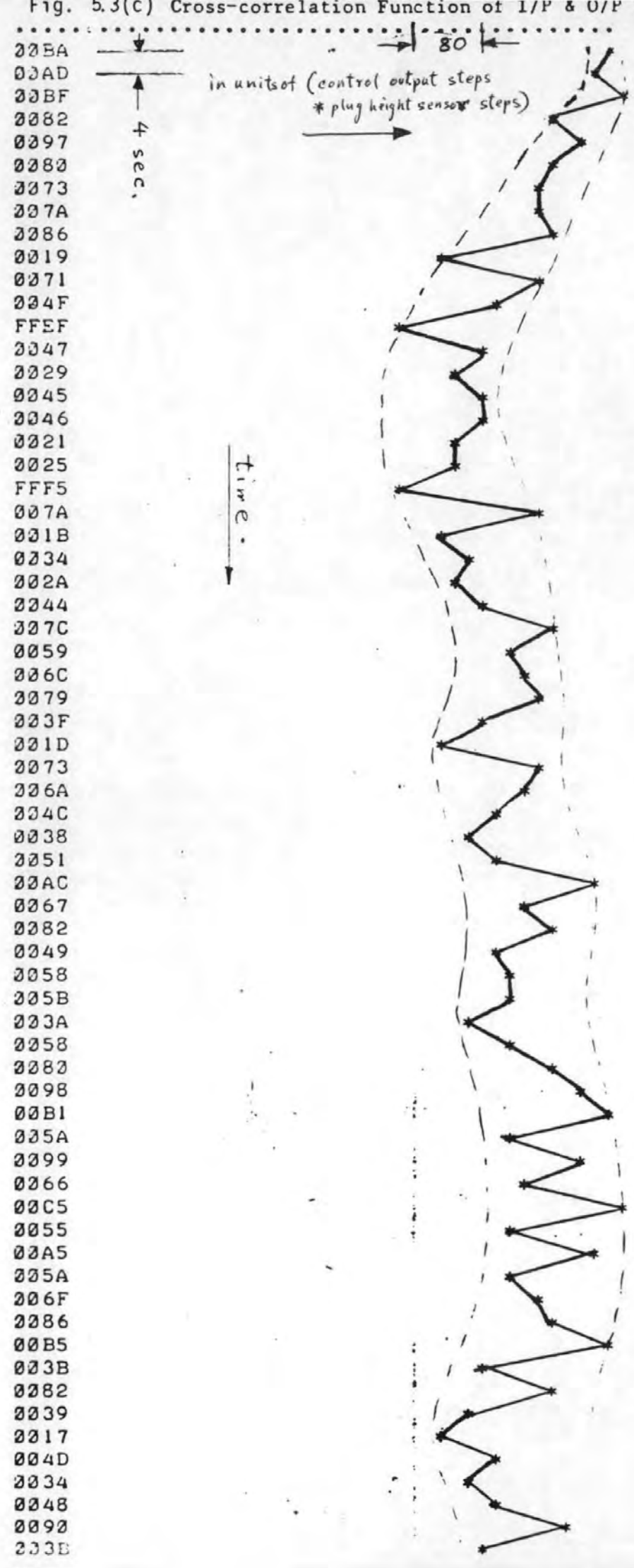


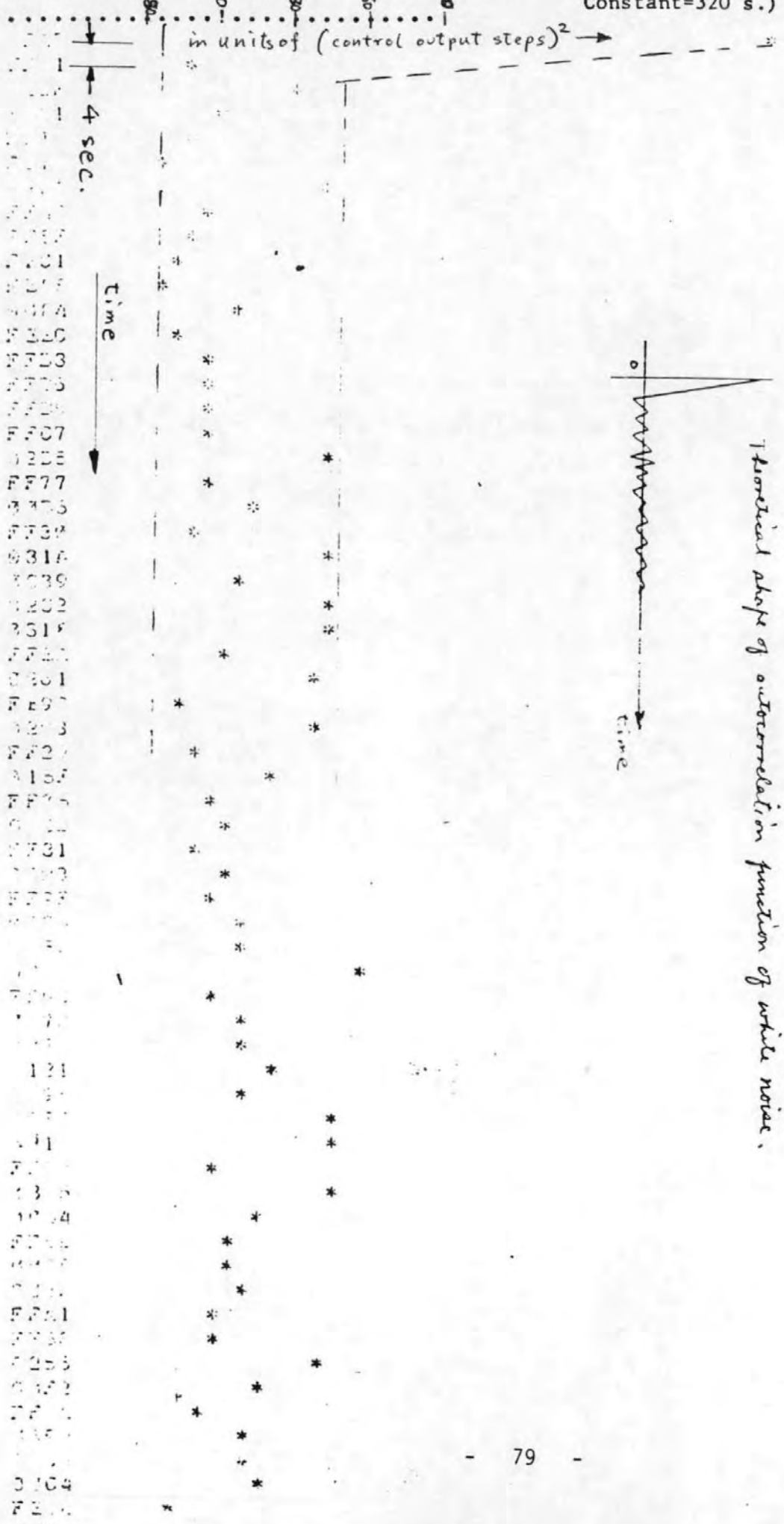
Fig. 5.3(b) Auto-correlation of input of fig. 5.3(a).

Fig. 5.3(C) Cross-correlation Function of I/P & O/P (H.P. Filter Time Constant=320 s.)



00BA
00AD
00BF
0082
0097
0080
0073
007A
0086
0019
0071
004F
FFEF
0047
0029
0045
0046
0021
0025
FFF5
007A
001B
0034
002A
0044
007C
0059
006C
0079
003F
001D
0073
006A
004C
0038
0051
00AC
0067
0082
0049
0058
005B
003A
0058
0080
0098
00B1
005A
0099
0066
00C5
0055
00A5
005A
006F
0086
00B5
003B
0082
0039
0017
004D
0034
0048
0090
003B

Fig. 5.3(d) Auto-correlation of input signal (H.P. Filter Time Constant=320 s.)



Theoretical shape of autocorrelation function of white noise.

Fig. 5.3(e) **Cross-correlation Function with suitable input level.**

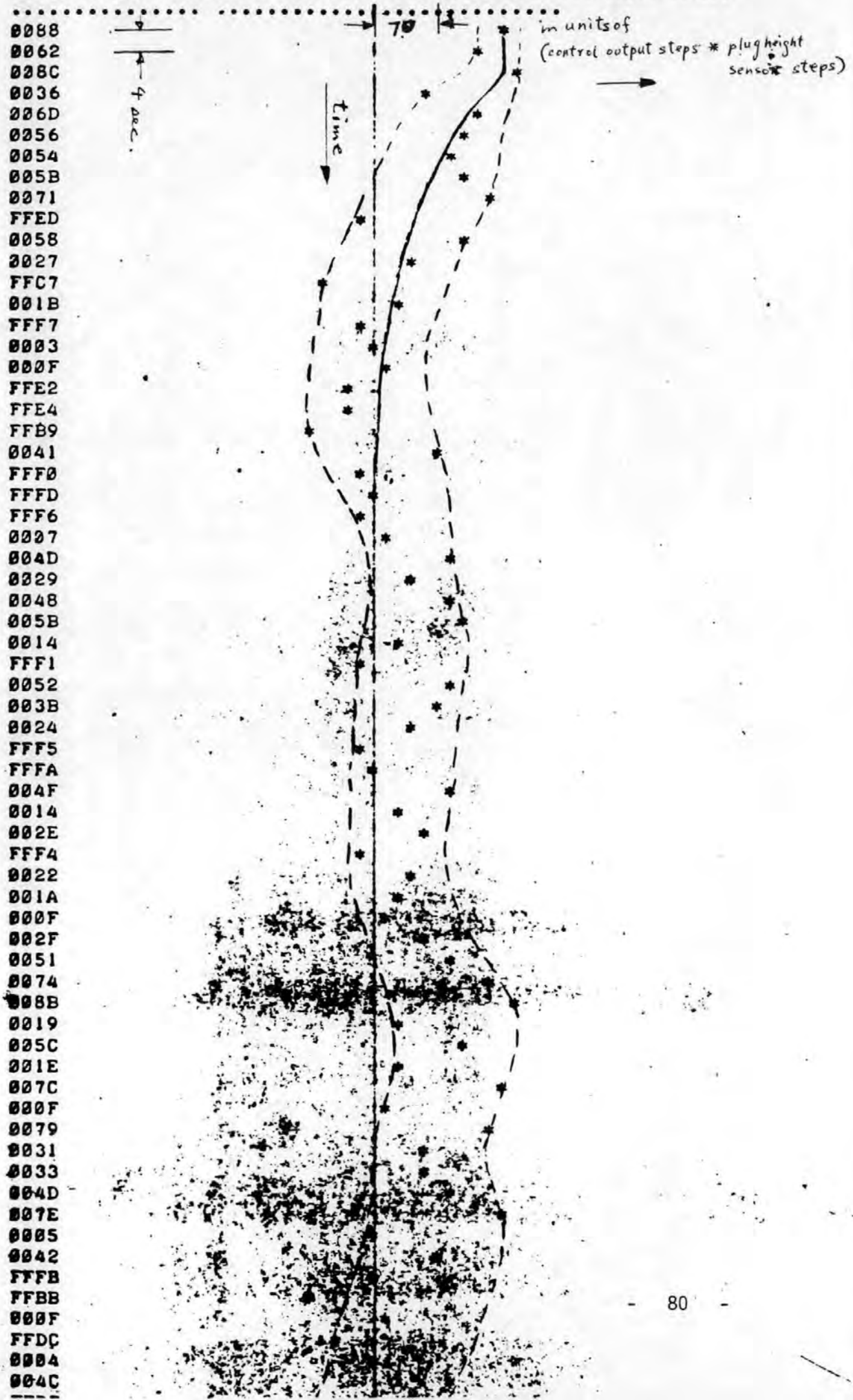


Fig. 5.3(f) Auto-correlation of input signal (H.P. Filter Constant=160 s.)

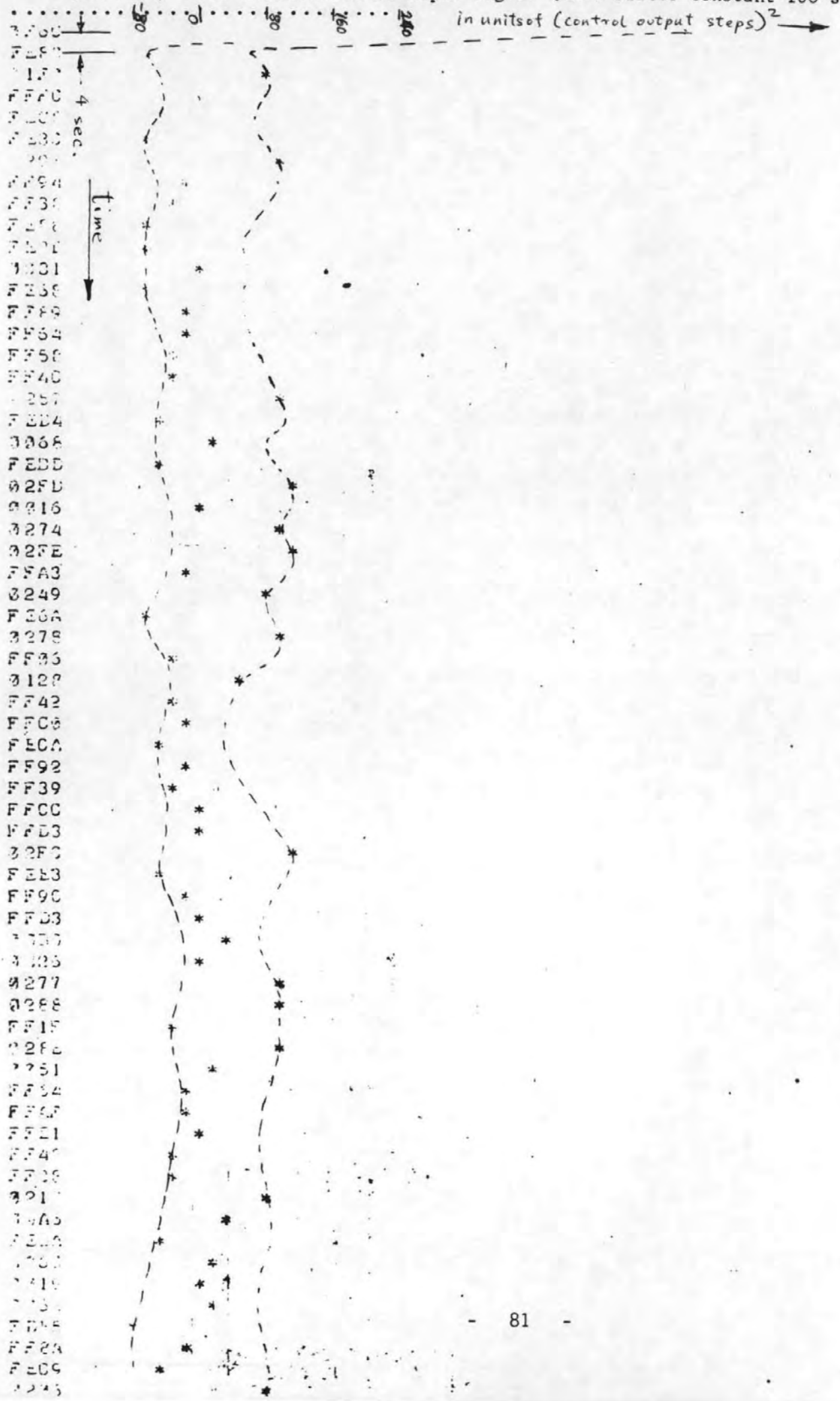
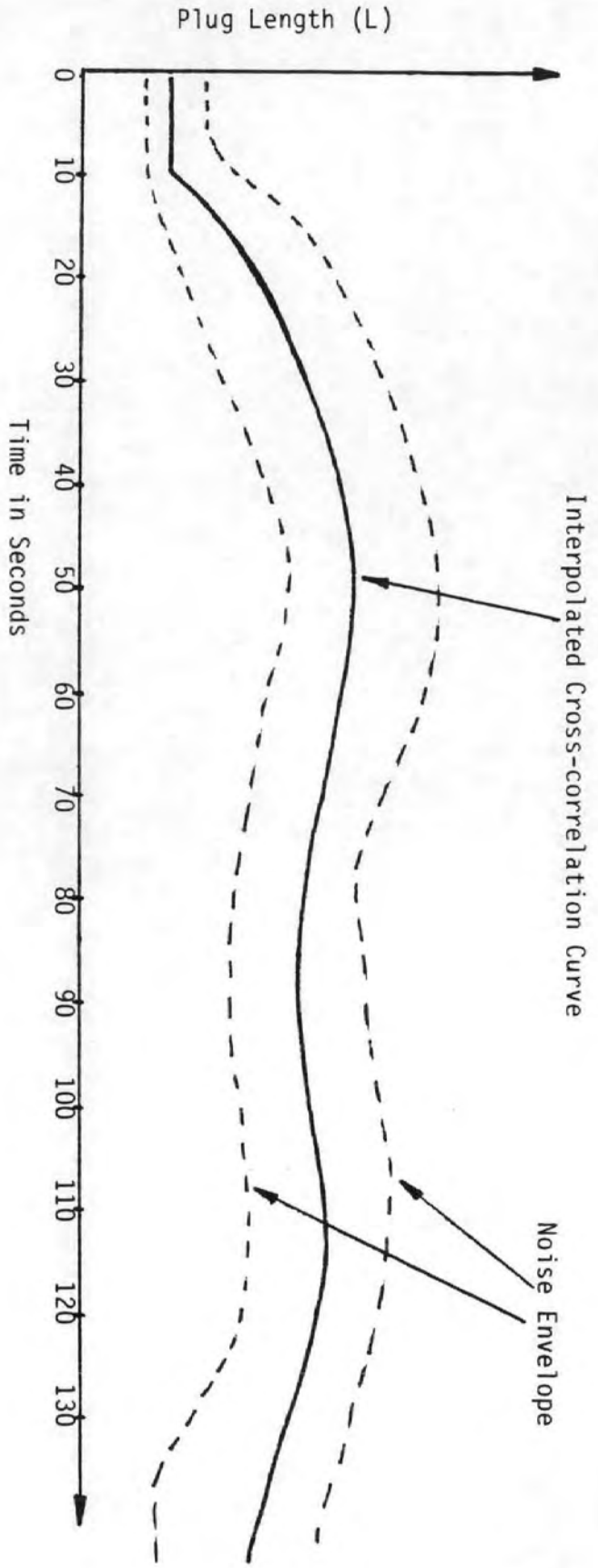


Fig. 5.3 (g) Cross-correlation Function of Plant I/P & O/P giving Open Loop Impulse Response (This is derived from Fig. 5.3 (e)).



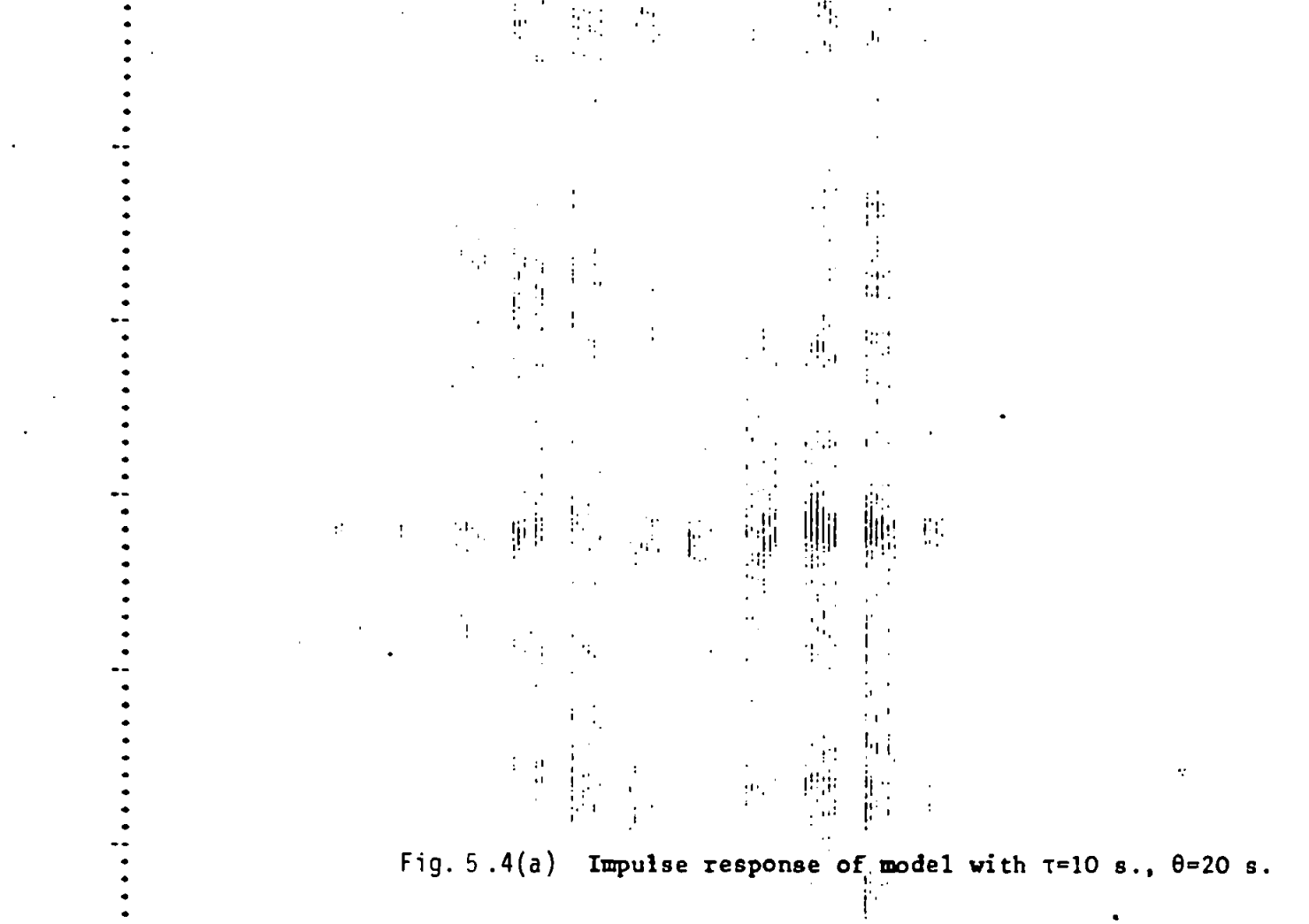


Fig. 5.4(a) Impulse response of model with τ=10 s., θ=20 s.

Fig. 5.4 (b) Impulse Response of (5.2.2)

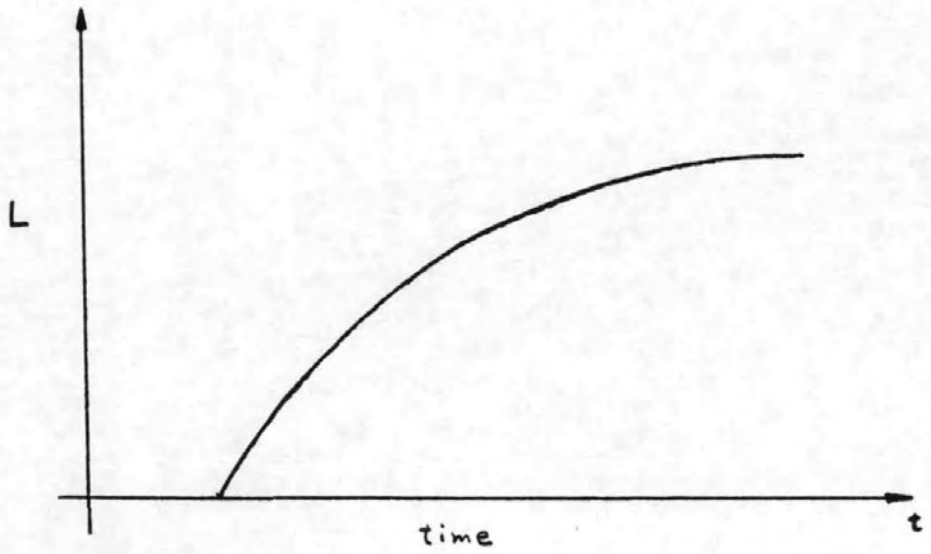
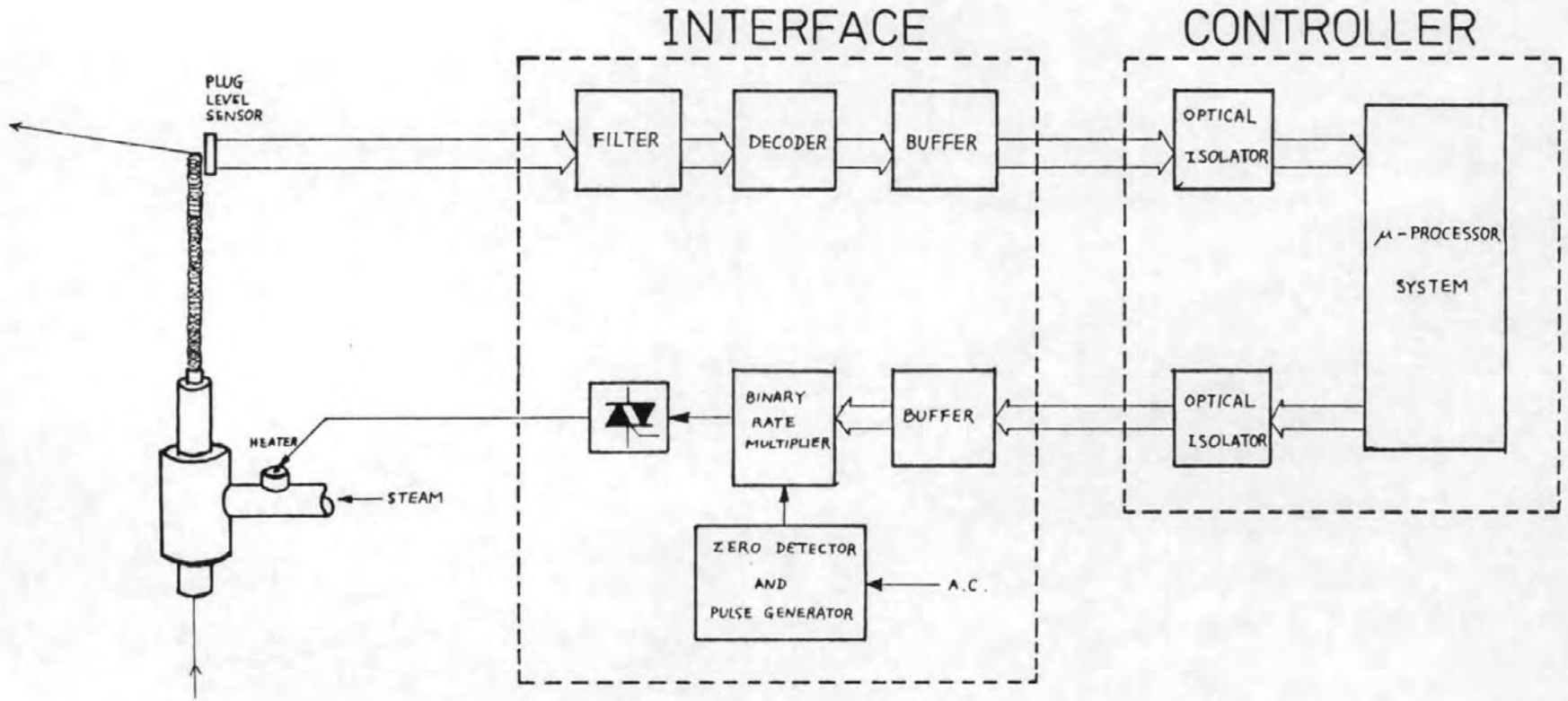


Fig. 5.5. Schematic Diagram of The Control Loop.



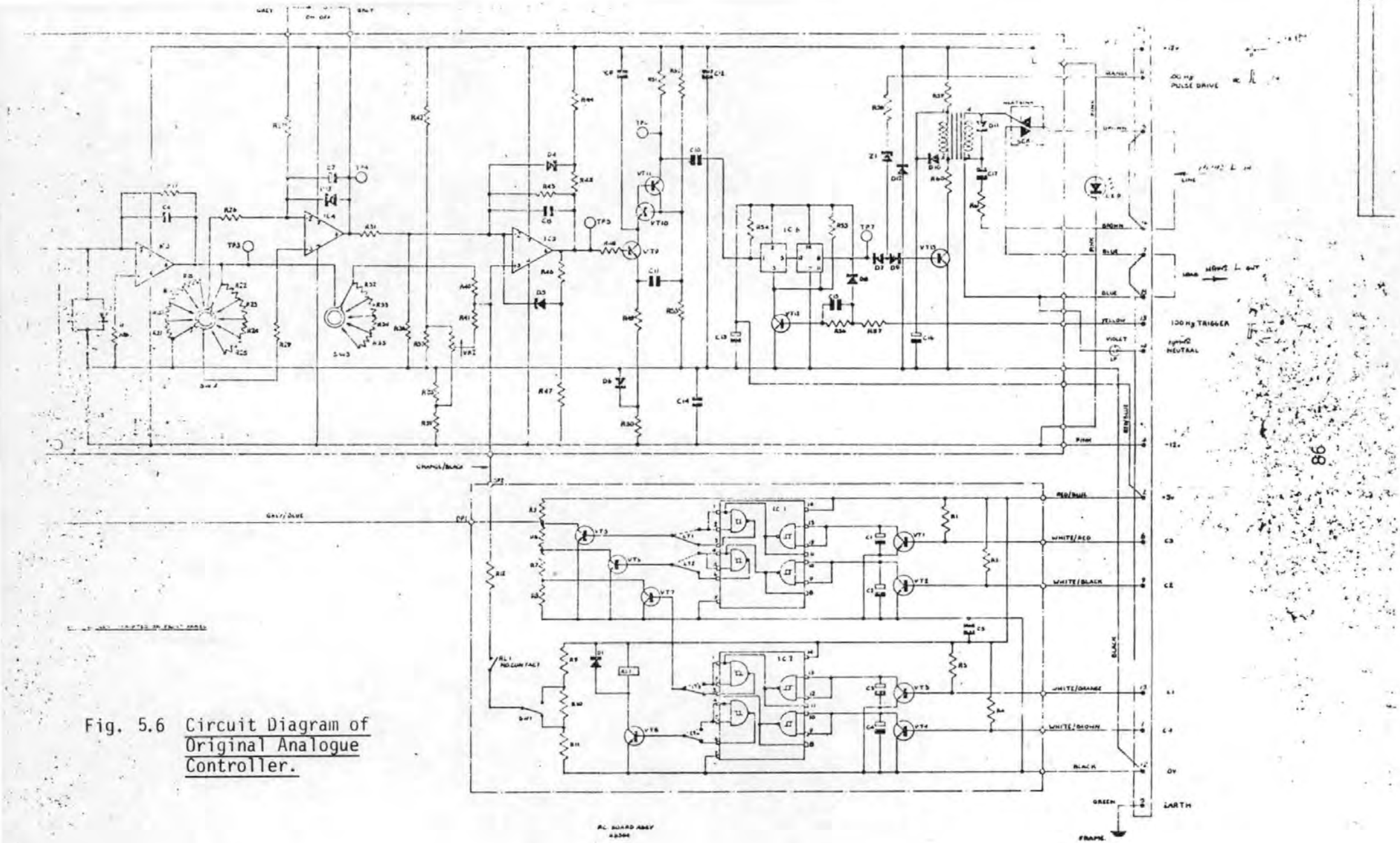


Fig. 5.6 Circuit Diagram of Original Analogue Controller.

Fig. 5.7 Temperature profile of a start up procedure controlled by the original analogue controller.

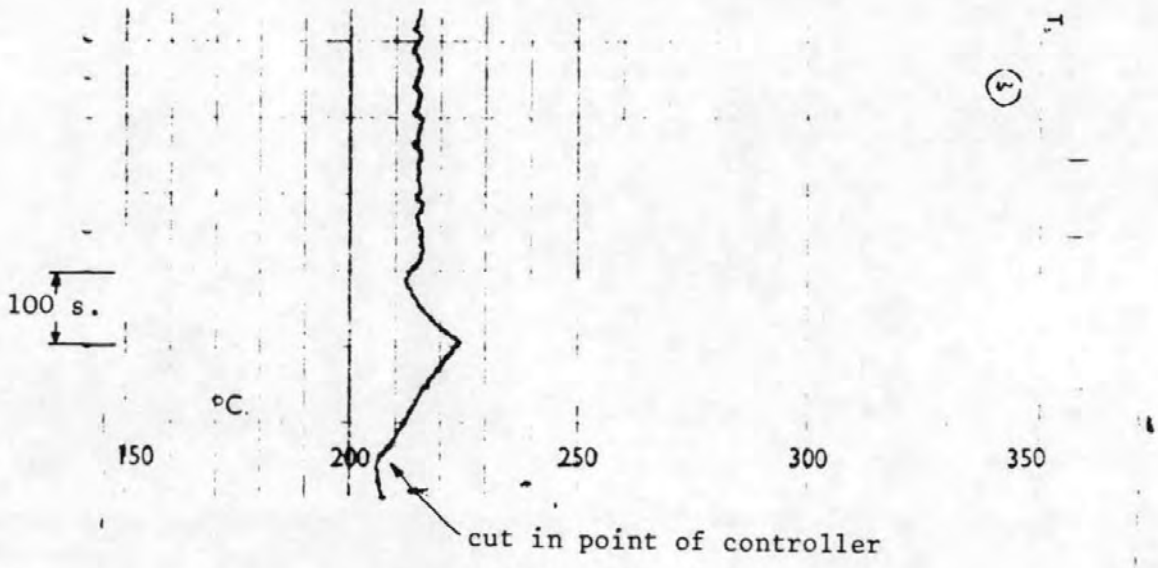


Fig. 5.8 Temperature profile of a start up procedure controlled by the Non-linear Adaptive controller.

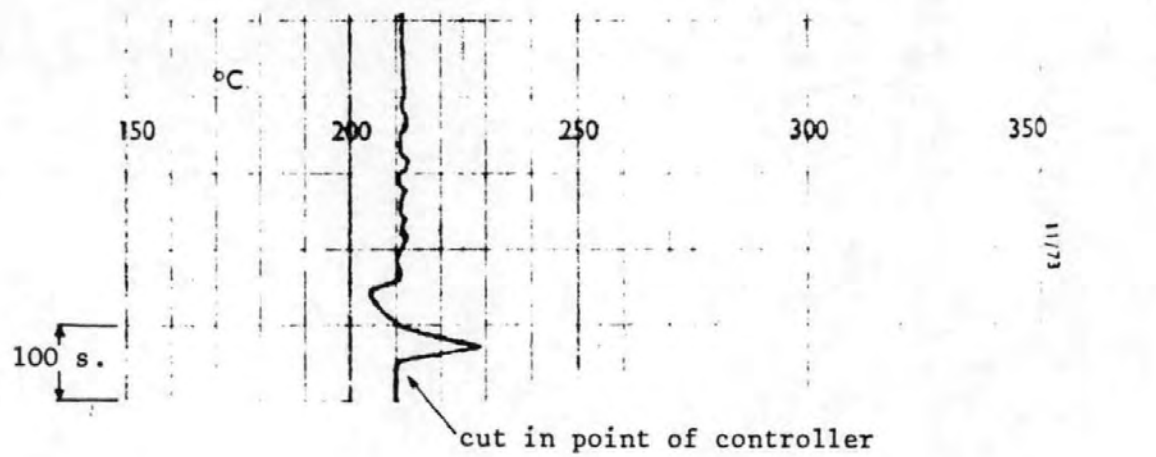
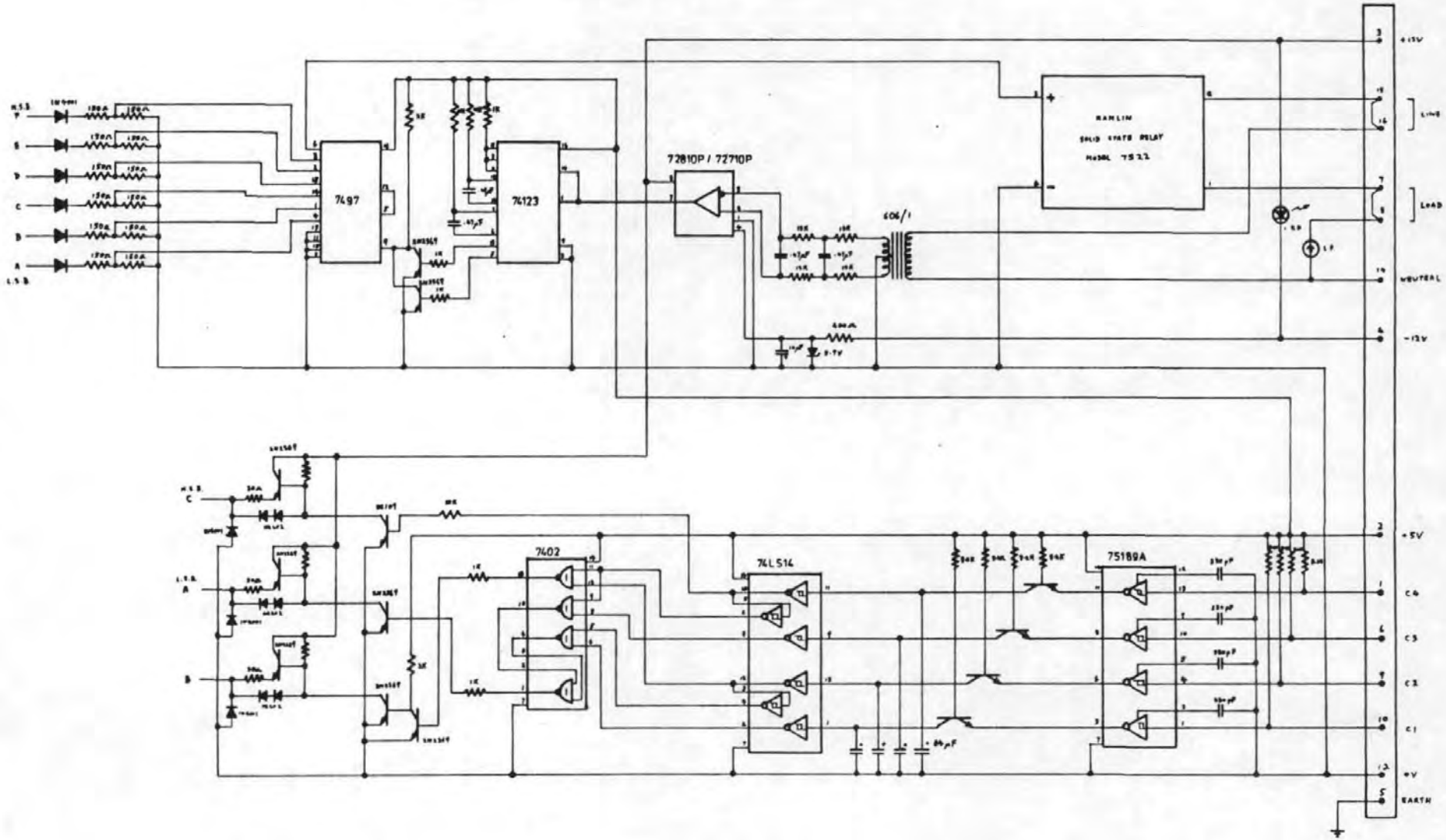


Fig. 5.9 **THE PLANT INTERFACE**



6. CONCLUSION

The advent of microprocessors in this decade has transformed the whole architecture of process control systems. Not only does it bring D.D.C. (direct digital control) to the single loop level, it also affects the basic structure of large process control systems which consists of a large number of control loops.

Only little more than a decade ago when D.D.C. was first applied to process control, it was largely implemented by main-frame computers in a simple monolithic arrangement as shown in Fig. 6.1 (ref. 1). Usually, one computer was used to control a whole plant despite the fact that computers were still not industrially proven. Therefore, a lot of problems arose due to computer failure which lead to complete plant shut down.

Later on, with the arrival of minicomputers, industrial processes were starting to use several computers in a distributed control arrangement while still maintaining the hierarchical structure as shown in Fig. 6.2.

Recently, with microprocessors invading the area of single loop controllers, a lot of processes have jumped from conventional analogue controllers to D.D.C. using microprocessors without going through the intermediate stages. This is due to the flexibility of being able to phase in the transition gradually by applying D.D.C. to the bottle neck loops first and gradually expanding it to all the loops as shown in Fig. 6.3 (a)

and (b). A computer can ultimately be installed to communicate and supervise all the loops as shown in Fig. 6.3 (c). This system may look more complex than the monolithic system. However, its advantages are: flexibility in implementation, simpler software and plant reliability less dependent on the supervisory computer.

With the ever increasing computing power of the microprocessors, especially with the arrival of 32 bit word machines, the development of local area networks and package switching protocols, future control system could become truly distributed. Individual controllers could simply be hooked onto a data highway. It can use its spare capacity to maintain the highway, cross-check other controllers and transmit cross-coupling information to other controllers in a multi-variable systems.

The non-linear adaptive controller developed in this thesis is a good example of a low cost advanced single loop microprocesor based controller working on a small industrial process. It has shown at the time that it is viable to develop low cost microprocessor based advanced controllers for small industrial process. As explained later in this chapter, new dedicated hardware has been designed to house the non-linear adaptive control algorithm for control of the plant. This dedicated hardware has provision for linking with other controllers. It provides a very good basis for future distributed control implementation. It provides a very useful framework on which we can base our future design.

It has been shown in chapter 5 that the non-linear adaptive controller has marked improvement over the conventional p & I controller in controlling the nylon crimping plant. However, due to the non-linearity at the output of the plant, it is not possible to prove that this controller is optimal. Obviously, changing the instrumentations on the plant may be the answer but, without any guarantee of substantial improvement to the process response it was difficult to justify the high cost.

The Algorithm Development/Implementation System developed for this project was found to be extremely useful both on and off site. Its multi-tasking design makes its functions virtually feels like working in parallel. It eased the burden a lot in the collection of data in plant trials, analysis of data and in algorithm development both in the college and on-site. It certainly had commercial potential at the time.

After the successful development of the non-linear adaptive control algorithm for the nylon crimping plant as discussed in chapters 4 and 5 a dedicated microprocessor based controller was designed to house the control algorithm. The circuit diagram of this dedicated controller is as shown in Fig. 6.4. Its design objective was to keep the circuit as simple as possible for low cost implementation and ease of maintenance. The circuit was packaged on a single printed circuit board the same size as that of the plant interface board to reside next to it in the same rack. The rack has the same physical dimensions

as that of the original analogue controller so that a direct replacement of the analogue controller can be achieved simply by unplugging the analogue controller and plugging in the microprocessor based controller. It also has provisions for linking with other controllers to perform multivariable control of several loops. The hardware of this dedicated controller is as shown in Fig. 6.5.

The nylon crimping plant is a very interesting process for studying the extension of the controller to multi-input/multi-output applications. Future development of the plant involves incorporating the drawing process as shown in the schematic diagram in Fig. 6.6.

Let the speed of the take up roller (1) be ω_1, and, the speed of drawing rollers (2) and (3) be ω_2 and ω_3 respectively. The system descriptions becomes:

$$\bar{x}(k+1) = A\,\bar{x}(k) + B\,\bar{u}(k) + \bar{\omega}(k)$$

$$z(k) = h^T\,\bar{x}(k) + d^T\,\bar{u}(k) + \bar{v}(k)$$

Where A, B are 4 X 4 matrices

and h, d, $\bar{x}$, $\bar{u}$, $\bar{\omega}$, $\bar{v}$ are 4th order vectors.

ω_1, ω_2 and ω_3 are subjected to the constraints:

$$s_1 \leq \omega_1 \leq s_1 + \mu$$

$$r_1 < \frac{\omega_1}{\omega_2} < r_1 + \delta$$

and $$\frac{\omega_2}{\omega_3} = r_2$$

Where the take up roller must be maintained at speed s_1 with

allowable variance μ and the ratio of the roller (1) to roller (2) is r_1 with variance δ. The drawing rollers must be kept at a constant speed ratio of r_2 between rollers (2) and (3) to obtain a constant drawing ratio.

From the above analysis, it seems that the system is not really a homogeneous multi-input/multi-output system but one which can be separated into several single loops with inter-coupling between them. Hence, it can be controlled by several single input/output controllers with coupling informations transmitted between them.

With this problem in mind, future development of the algorithm in multi-variable applications may be in the form of several individual single input/output controllers each controlling one loop while at the same time linked together to decouple interaction between control loops in multi-input/multi-output feedback control system as shown in Fig. 6.7. This format of application limits the system to only transmission of coupling information between control loops while falling short of full multi-variable control. However, in practical applications, this limited implementation could have solved most control problem as explained by Shinskey (Ref. 34).

The above dedicated controller was designed with a specific application in mind, it is not typical of microprocessor based controllers. The hardware structure of a generalized microprocessor based controller should include features such as a

watch dog timer, automatic fault diagnosis, communication interfaces (for peripherals and local area networks) and a functional front panel which provides display of vital parameters, choice of control algorithms and modification of tuning characteristics. The basic hardware structure of a generalized microprocessor based process controller is as shown in the schematic diagram in Fig. 6.8.

With numerous new control algorithms being proposed nowadays and more to come in the future, each of them claiming to be the best, perhaps rightly so in certain applications, the ultimate choice of control algorithms has to rest on the control engineer. Therefore, any new control algorithm developed should be presented in such a way that it is easily understood by control engineers. Furthermore, future design of microprocessor based process controllers should allow for a choice of several control algorithms and the flexibility to include new algorithms, so that control engineers can try each of them in turn in order to find the best fit.

FIG. 6.1 Simple Monlithic System

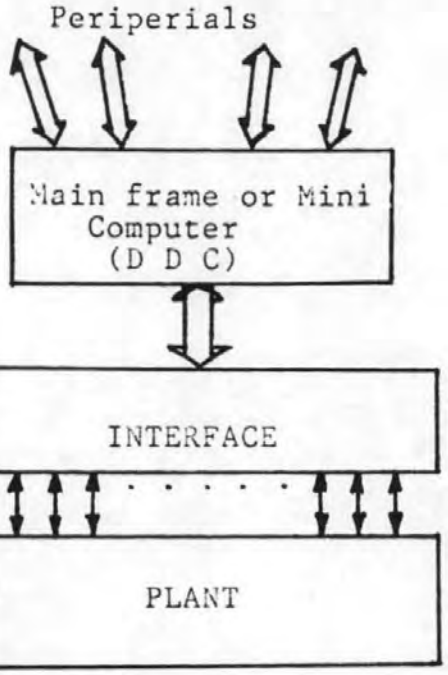


FIG. 6.2 Hierarchical System

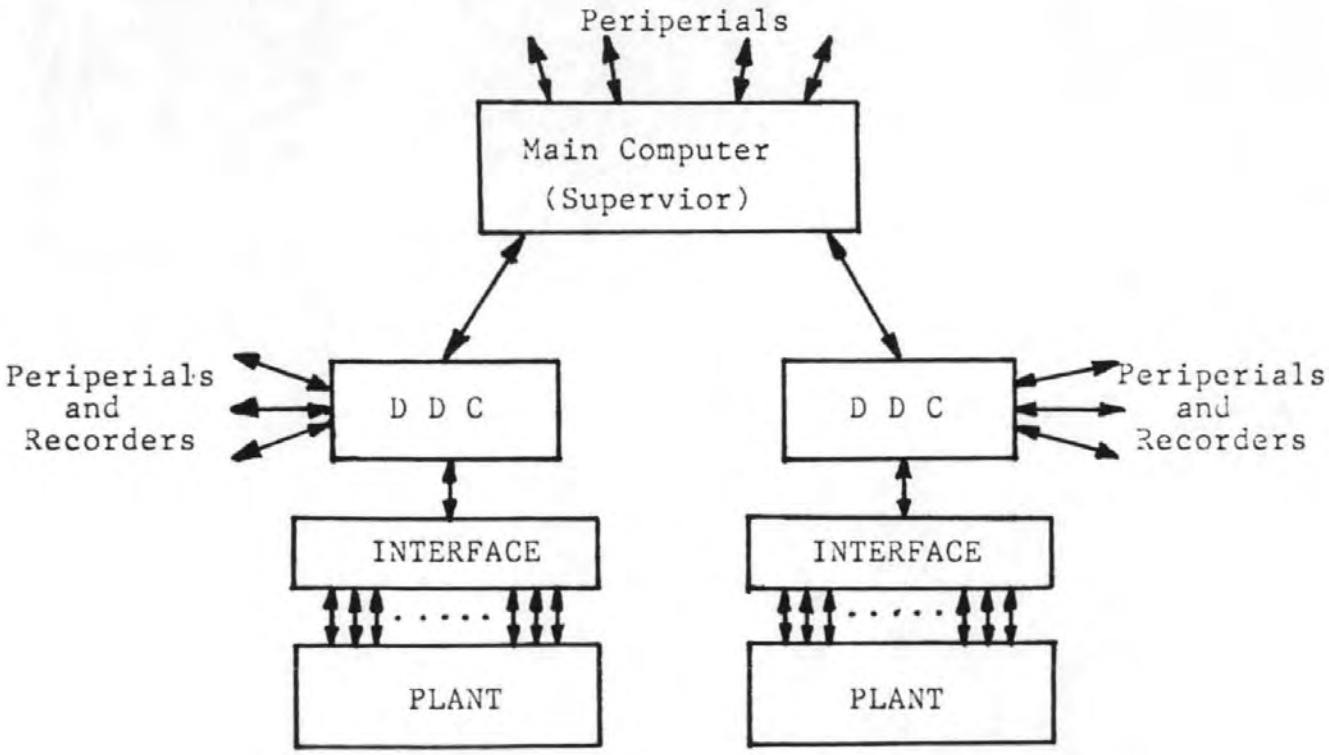
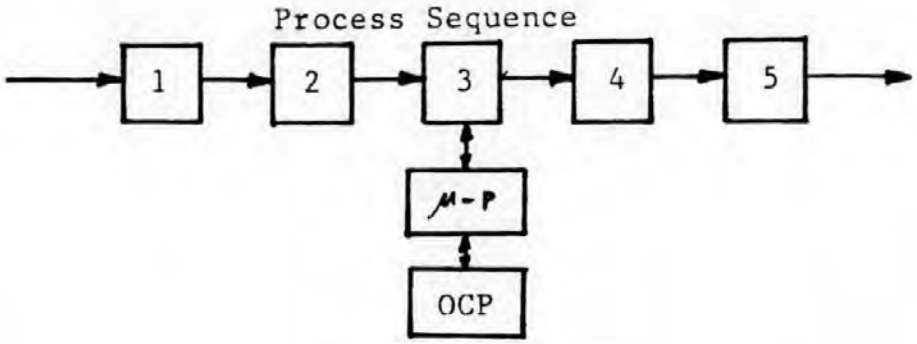


FIG. 6.3(a) Use Intelligent Controller to improve bottle neck.



OCP = Operator's Control Panel.

FIG. 6.3(b) Expand Gradually to include all loops.

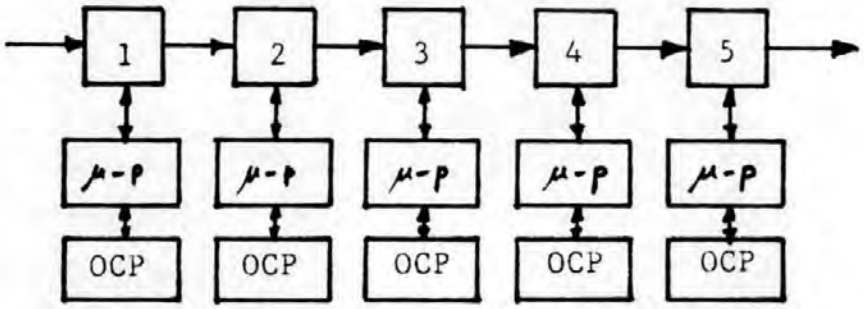


FIG. 6.3(c) Distributed Control with Central Supervisor.

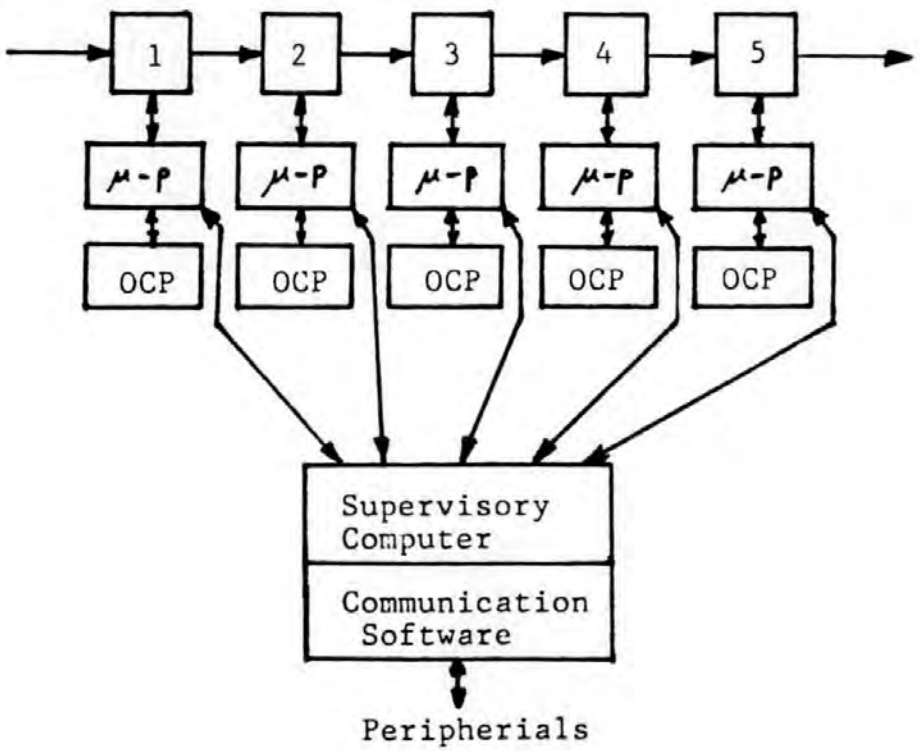


Fig. 6.4. Circuit Diagram of Prototype Controller.

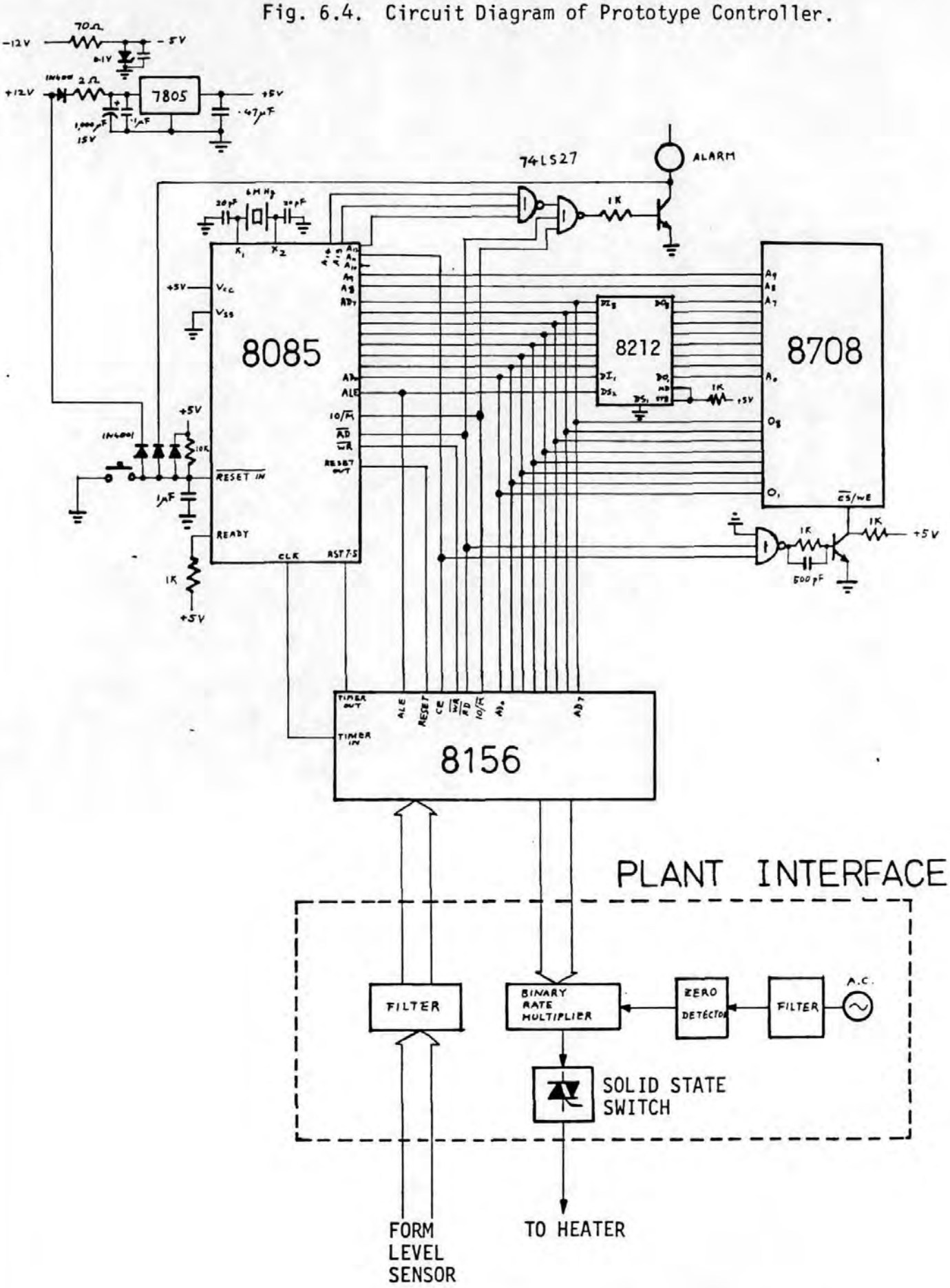
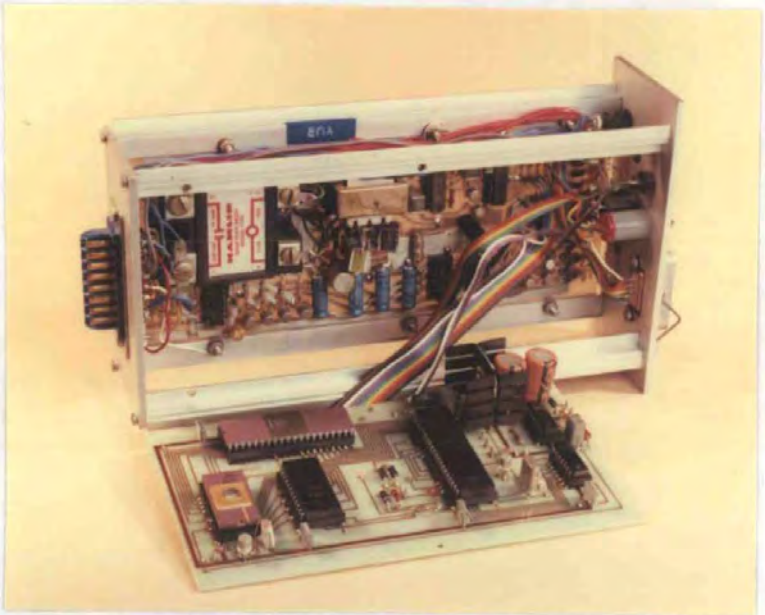


Fig. 6.5 Hardware Layout of Dedicated Controller



PLANT INTERFACE WITH THE MICROPROCESSOR BASED CONTROLLER BOARD BY ITS SIDE

ASSEMBLY OF DEDICATED CONTROLLER

Fig. 6.6 Schematic Diagram of the 2nd phase Plant.

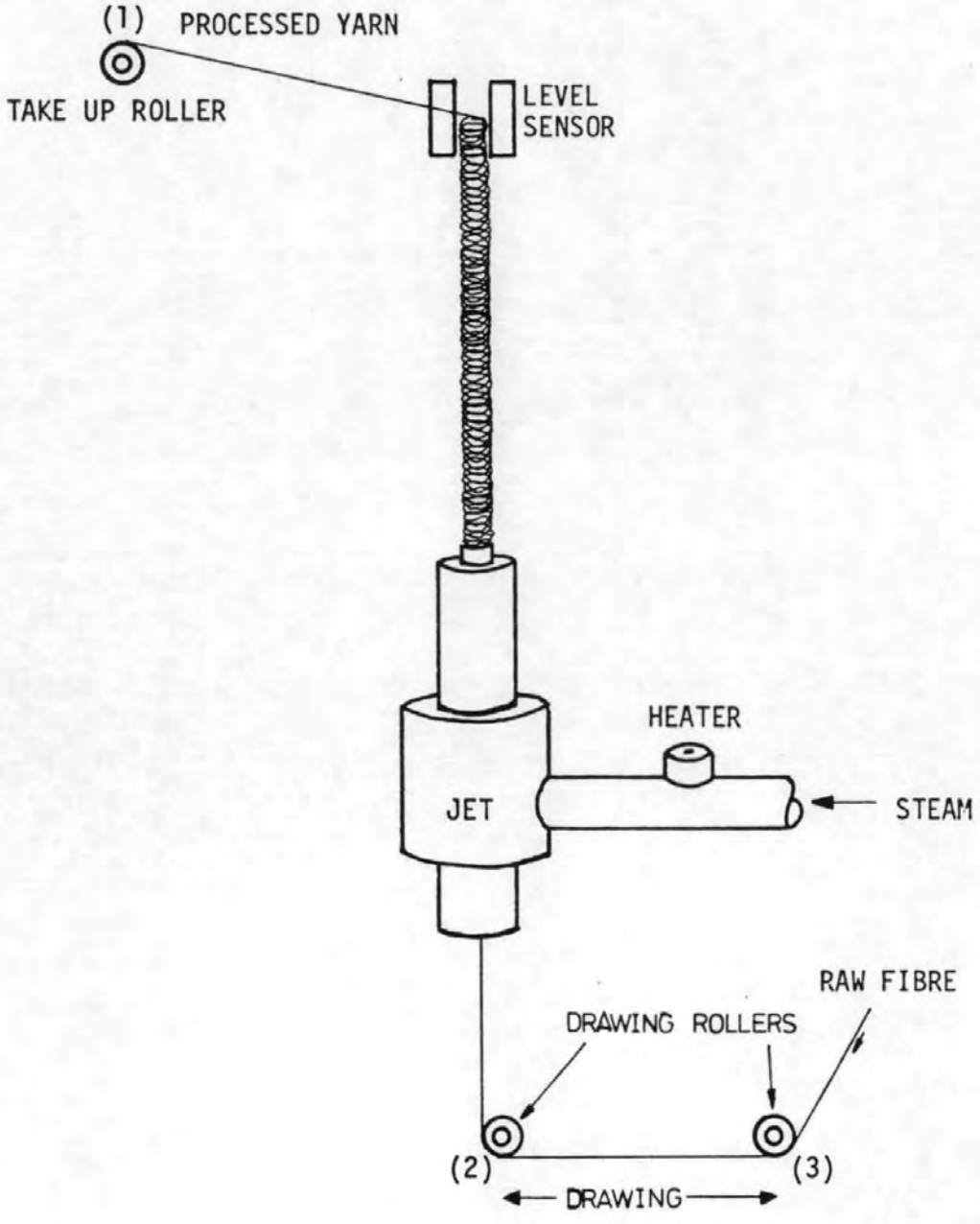
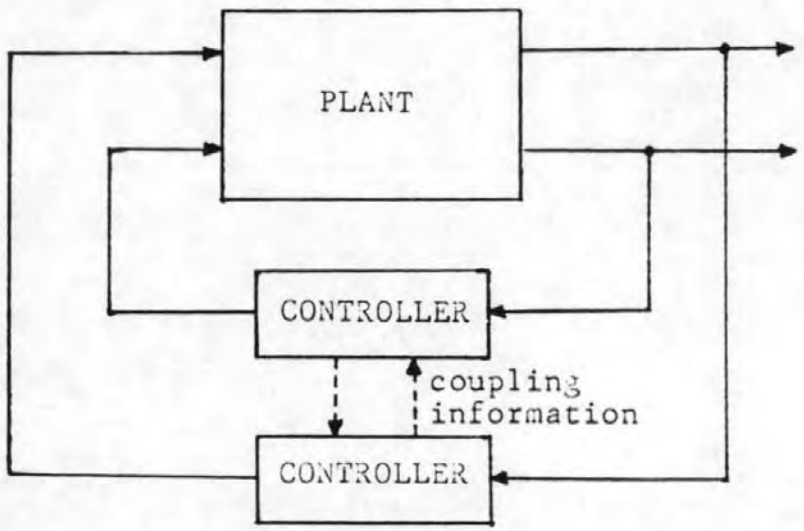


Fig. 6.7 Generalised Self-tuning Controller Application in Multi-input/Multi-output Feedback Systems.

(a) Simple transfer of coupling information.



(b) Separate controllers for decoupling and control.

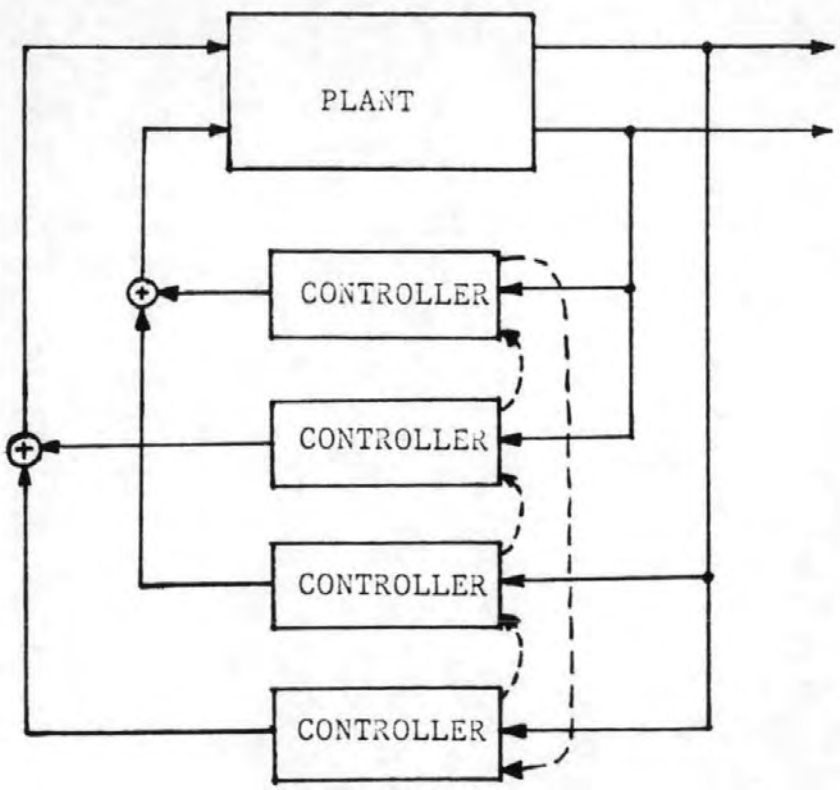
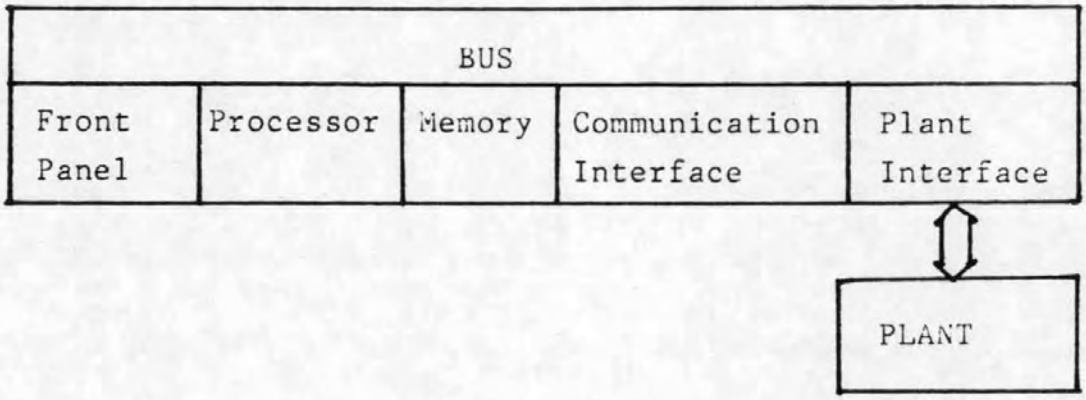


FIG. 6.8 Schematic Diagram Showing the Hardware Structure of a Generalised Microprocessor Based Process Controller.



R E F E R E N C E S

(1) CARTRIGHT L.M. and MAYBERRY J.C. "Distributed Systems - A New Approach to Process Control" - Paper No. 11; ISA/75 Annual Conference; October, 1975.

(2) FRAADE D.J. and GAST S. "A Survey of Computer Network and Distributed Control". Proceedings of the 5th IFAC/IFIP International Conference. The Hague, the Netherlands, June 14-17, 1977.

(3) WHITAKER, H.P. "Design Capabilities of Model - Reference Adaptive Systems", Proc. Natl Electron. Conf., 18, 241-249 (1962).

(4) OSBURN, P.V., WHITAKER, H.P., and KEZER, A., "New Developments in the Design of Model Reference Adaptive Control Systems", Institute of Aerospace Sciences, Paper No. 61-39, Jan. 1961.

(5) PARKS, P.C., "Lyapunov Redesign of Model Reference Adaptive Control System", IEEE Trans. Automatic Control, AC-11, (3), 362-367 (1966).

(6) GIBSON, J.E., "Non-linear Automatic Control", McGraw New York, 1962.

(7) GRAHAM, D., and LATHROP, R.C., "The Synthesis of Optimum Transient Response, Criteria and Standard Forms", Trans. AIEE, 72, 273-288 (1953).

(8) LARDAU, I.D., "A Survey of Model Reference Adaptive Techniques - Theory and Applications". Automatica, 1974, 10, pp 353-379.

(9) NARENDRA, K.S., and VALANVANI, L.S., "Stable Adaptive Observers and Controllers", Proc. IEEE, 1976, 64, (8), pp 1198-1208.

(10) BELLMAN, R., "Adaptive Processes - A Guided Tour", Princeton University Press, Princeton N.J. 1961.

(11) MEIER, L., LARSON, R.E. and TETHER, A.J., "Dynamic Programming for Stochastic Control of Discrete Systems", IEEE Trans. Automatic Control AC-16, 767-775, (1971). Special issue on Linear Quadratic Gaussian Problem.

(12) TOU, J., and JOSEPH, P., "On Linear Control Theory", Pt. II, AIEE Trans. 80, 18, (1961).

(13) LEE, R.C.K.,"Optimal Estimation Identification and Control" MIT Press, Cambridge, Mass., 1964.

(14) SARIDIS, G.N. and LOFFIA, R.N., "Parameter Identificaiton and Control of Linear Discrete-Time Systems", IEEE Trans. Automatic Control, AC-17, (1) 52-61 (1972).

(15) CLARKE D.W., COPE, S.N., GAWTHROP, P.J., "Feasibility Study of the Application of Microprocessors to Self-tuning Controllers", University of Oxford, Department of Engineering Science Report. No. 1137/75 (1975).

(16) NARENDRA K.S., PETERSON, B.B., "Recent Development in Adaptive Control", (Yale Univ., New Haven, C7, U.S.A.) March, 1980.

(17) DREIFKE, R.M., "An Approach to Model-Referenced Adaptive Control System", IEEE Trans. Automatic Control, AC-12, (1), 75-80 (1967).

(18) DYMOCK, A.J., MEREDITH, J.F., HALL, A., and WHITE, K.M., "Analysis of a Type of Model Reference Adaptive Control System", Proceedings of IEEE, 112, (4), 743-753 (1965).

(19) EVELEIGH, V.W., "Adaptive Control and Optimization Techniques", McGraw-Hill, New York, 1967.

(20) WESTERLAND T., TOIVONEN H., NYNAN K.E., "Stochastic Modelling and Self-tuning Control of a Continuous Cement Raw Material Mixing System", (Dept. of Chemical Engng., Abo Akademi, Abo, Finland), Proceedings of the 18th IEE Conference on Decision and Control including the Symposium on Adaptive Process. Pt. I, Fort Lauderdale, Fl, USA, 12-14 Dec., 1979 (New York, U.S.A.: IEEE 1979) Pg. 610-15.

(21) VASUDERA K.S., MANN W.S., "Design of a Self-tuning Regulator for Process Control Applications", (Dept. of Electrical and Electronic Engrg., James Cook Univ. of North Queenland, Queenland, Austrlia), Australian Conference on Control Engrg. Melbourne, Australia, 5-7 June, 1979, (Barton, Australia: Instn. Engrs. Australia 1979), p. 117-21.

(22) JOHNSTONE, R.M., FISHER, D.G., SHAH, S.L., "Hyperstable Adaptive Control - An Indirect Input-output Approach with Expicit Model Identification", (Dept. of Chem Engrg., U. of Alberta, Edmonton, Alberta, Canada.) Proceedings of the 1979 Joint Automatic Control Conference, Denver, Co., U.S.A., 17-21 June 1979 (New York, U.S.A.: American Inst. Chem. Engrs. 1979) p. 487-94.

(23) SARIDIS, G.N., "On a Class of Performance - Adaptive Self-Organizing Control Systems", In Pattern Recognition and Machine Learning (K.S. Fu, ed.), Plenum Press, New York, 1971.

(24) SARIDIS, G.N., "Expanding Subinterval Random Search for System Identification and Control, "Proc. IV IFAC Symp. Identification, etc. Tbilisi GSSR, Sept. 1976.

(25) YUNG, K.L., "Kalman Filter for 3-term Control", M.Sc. thesis, Imperial College of Science & Technology (Univ. of London), London, Sept. 1976.

(26) CLARKE, D.W.,"Introduction to Self-tuning Controllers", Chapter 2, Self-tuning and Adaptive Control: Theory and Applications (C.J. Harris & S.A. Billings, edit.), IEE Control Engrg. Series 15 (1981).

(27) SARIDIS, G.N., "Comparison of Six On-line Identification Algorithms", Automatica, 10, (1) 69-80 (1974).

(28) ISERMANN, R., et al, "Comparision of Six On-line Identification and Parameter Estimation Methods with Three Simulated Processes", Automatica, 10, (1) 81-104 (1974).

(29) EYKHOFF Pieter, "System Identification, Parameter and State Estimation". London, Wiley (1977).

(30) EYKHOFF Pieter, "Trends and Progress in System Identification", Pergamon Press. Oxford Pergamon Press (1981).

(31) EVERETT D. (1966) "Periodic Digital Sequences with Pseudonoise Properties" G.E.C. Journal, Vol. 33, No. 3., p. 115-126.

(32) MARITSAS D.G. (1973) "The Autocorrelation Function of the Two Feedback Shift Register Pseudorandom Source", IEEE Transactions, C22, p. 962-964.

(33) HAMMING R.W. "Numerical Methods for Scientists and Engineers", McGraw-Hill, N.Y. 1962, pages 34 and 389.

(34) SHINSKEY F.G. (1977) "The Stability of Control Loops with and without Decoupling", IFAC Conference on Multivariable Technological Systems, University of New Brunswick, July 5th-8th, pp. 221-230.

(35) TORODE J., "Cost-Effective Networking" (Digital Microsystem Ltd., Twyford, England.) Computer System (GB), vol. 3, No. 3, p.34-40 (March 1983).

(36) MUMMERLE K., REISER M. "Local Area Communication Network - An Overview", (IBM Zurich Res. Lab., Zurich, Switzerland.) J. Telecommun. Network (U.S.A.) vol. 1, No. 4, p. 349-70 (1982).

(37) GROSS, S.R., HINRIKSEN U., STIEGE G., "COSY - A Local Computer Network Based on An Autonous Data Communication System" (Lehrstuhl D fur Informatik, Tech. Univ. Braunschweig, Braunschweig, Germany.) Angew. Inf. (Germany), vol.25, No. 2, p. 62-7 (Feb. 1983).

(38) METCALFE R.M., BOGGS D.R., "Ethernet: Distributed Packet Switching for Local Computer Network", (Xerox Palo Alto Res. Centre, Palo Alto, CA, USA.), Jan. 1983.

(39) DAHOD A.M., "Local Network Standards: No Utopia", (Applitek Corp., Woburn, M.A., U.S.A.) Data Commun. (U.S.A.) vol. 12, p.173-6, 179-80 (March 1983).

(40) CASEY D. "Why Xinet could be a giant among the Local Networks", Compu. Wkly. (GB) No. 872, p. 15 (11 Aug., 1983).

(41) MACMILLAN R.H. "Non-linear Control System", p. 340-345. Oxford Pergamon Press (1962).

(42) GRAHAM & McRUER "Analysis of Non-linear Control System", p. 340-345.

(43) THALER & PASTEL "Analysis and Design of Non-linear Feedback Control System", p. 89-90 and p. 310. Mcgraw Hill Electrical and Electronic Eng. Series (1962).

(44) TSE, E., & BAR-SHALOM, Y., "Concepts and Methods in Stochastic Control", in Control and Dynamic System, (C.T. Leondes, ed.) Academic Press, New York, 1975.

(45) JACOBS, O.L.R., "On Non-linear Adaptive Control" Report OUEL-1323/80 Univ. Oxford England (1980).

(46) GURIN, L.S., and RASTRIGIN, L.A., "Convergence of the Random Search Method in the Presence of Noise", Automation Remote Control, 25 (9), 1505-1511 (1965).

(47) SAGE A.P. and MELSA J.L. "System Identification", Mathematics in Science and Engineering. Academic Press (1971).

(48) WINSTON, A.W. and SMITH, T.B. "Use of the Z-80 in Data Collection and Control". (New IKOR Inc., Gloucester, MA, USA). IECI '78 Annual Conference Proceedings on Industrial Applications of Microprocessors, Philadelphia, PA, USA, 20-22 March 1978 (New York, USA: IEEE 1978), p. 208-14.

(49) LEE, Chin-Hwa, "A Multitask Executive System for a Microcomputer". Dept. of Electrical & Computer Engng., Syracuse Univ., Syracuse, N.Y., USA). Proc. of The International Symposium on Mini and Micro Computers, Montreal, Canada, 11-18 Nov. 1977.(New York, USA: IEEE 1978), P.74-83.

(50) CHIEN, Y.C., "Multitasking Executive Simplifies Realtime Microprocessor System Design". (Systems & Software Inc., Downers Grove, IL, USA) Comput. Des. (USA), vol. 19, No. 1, P. 109-17 (Jan. 1980).

APPENDIX 1

The derivation of the incremental 3-term control algorithm.

The continuous time 3-term controller is basically composed of integral, proportional and differential of the error signal as follows:

Controller output $C = Kp\left(\frac{1}{\tau_I}\int \varepsilon dt + \varepsilon + \tau_D \frac{d\varepsilon}{dt}\right)$

$$\frac{dC}{dt} = Kp\left(\frac{\varepsilon}{\tau_I} + \frac{d\varepsilon}{dt} + \tau_D \frac{d^2\varepsilon}{dt^2}\right)$$

$$\Delta C = \frac{dC}{dt}\tau_S = K_p\left(\frac{\tau_S}{\tau_I} + \tau_S \frac{d\varepsilon}{dt} + \tau_D\tau_S \frac{d^2\varepsilon}{dt^2}\right)$$

Where ε = error

τ_s = sample interval

and Kp, τ_I, τ_D are constants.

The approximation of $\frac{d\varepsilon}{dt}$ and $\frac{d^2\varepsilon}{dt^2}$ are as follows:-

$$\frac{d\varepsilon}{dt} = \frac{\varepsilon_t - \varepsilon_{t-1}}{\tau_s} = \frac{\Delta\varepsilon}{\tau_s}$$

$$\frac{d^2\varepsilon}{dt^2} = \frac{1}{\tau_s}\left(\frac{d\varepsilon}{dt_t} - \frac{d\varepsilon}{dt_{t-1}}\right) = \frac{\Delta^2}{\tau_s^2}$$

$$= \frac{\varepsilon_t - 2\varepsilon_{t-1} + \varepsilon_{t-2}}{\tau_s^2}$$

Hence the incremental 3-term controller algorithm becomes:

$$C = Kp\left(\frac{\tau s}{\tau_I}\varepsilon + \Delta\varepsilon + \frac{D}{\tau_S}\Delta^2\varepsilon\right)$$

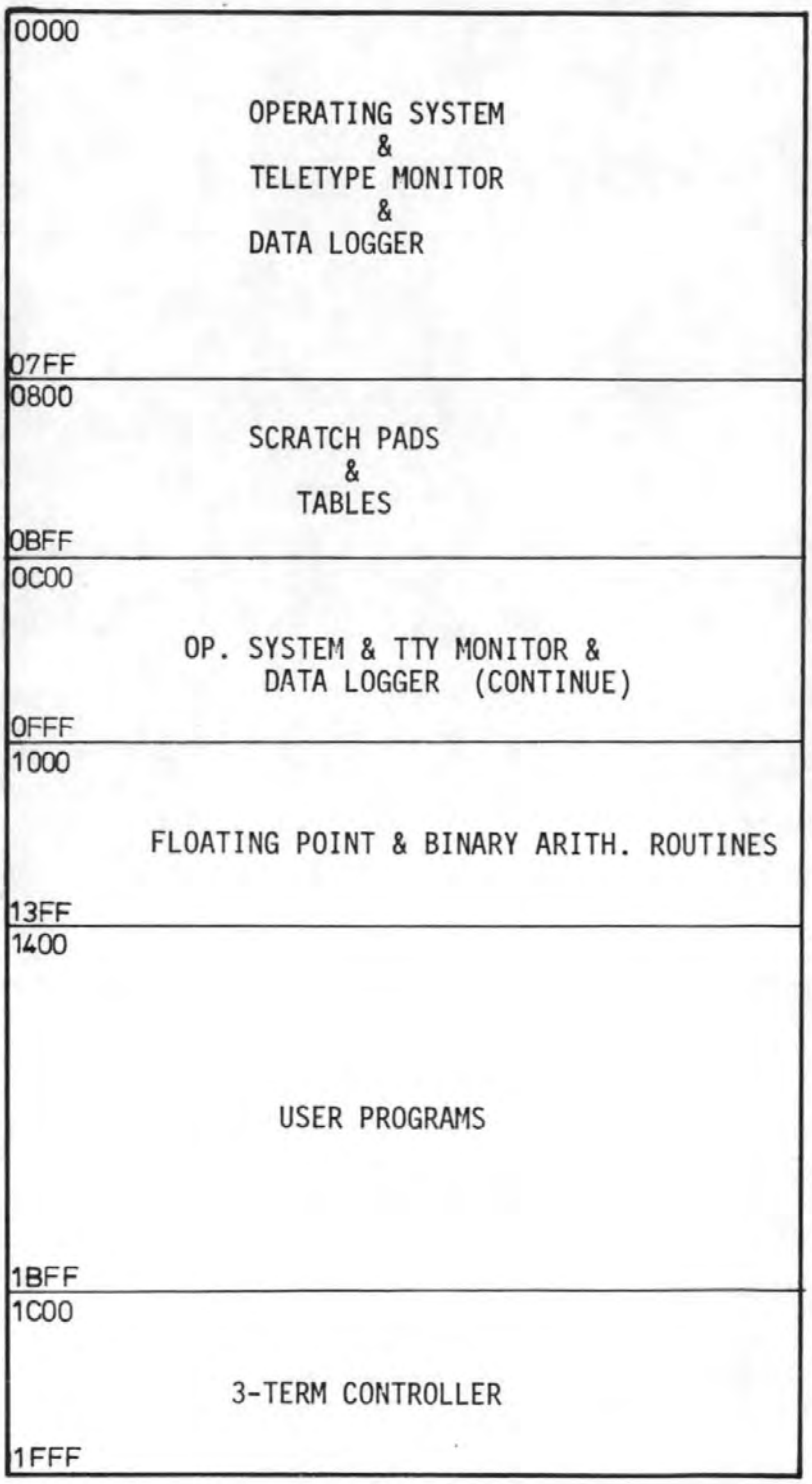
$$= \left(K_I\tau_S\varepsilon + Kp\Delta\varepsilon + \frac{K_D}{\tau_S}\Delta^2\varepsilon\right)$$

$$C_t = C_{t-1} + \Delta C_t$$

APPENDIX 2

1. Memory Map of the 'Algorithm Development / Implementation System'.

(A) Major Sections:



(b) Scratch Pads and Tables:

0800 CONSTANT TABLE 092B	
092C	O/P BUFFER POINTER FOR TX 1
092E O/P BUFFER FOR SERIAL TRANSMISSION INTERFACE 1 (TX 1) 098F	
0990	O/P BUFFER POINTER FOR TX 2
0992 O/P BUFFER FOR SERIAL TRANSMISSION INTERFACE 2 (TX 2) 09F3	
09F4 BUFFER FOR "PSW,A,C,B,E,D,L,H,SPL,SPH" (FOR 'TRACE') 09FD	
09FE	STACK POINTER BUFFER (FOR 'TRACE')
0A00	INITIAL PROGRAM COUNTER BUFFER 1 (FOR 'TRACE')
0A02	TERMINAL PROGRAM COUNTER BUFFER (FOR 'TRACE')
0A04	INSTRUCTION BUFFER (FOR 'TRACE')
0A07	'JUMP BACK' (TO 'TRACE' PROGRAM) INSTRUCTION
0A0A	INITIAL PROGRAM COUNTER BUFFER 2 (FOR 'TRACE')
0A0C	'N' (O/P INHIBIT) REGISTER (FOR 'TRACE')
0A0D	INSTRUCTION LENGTH (NUMBER OF BYTE) (FOR 'TRACE')
0A0E	BUFFER FOR PROGRAM COUNTER (FOR O/P) (FOR 'TRACE')
0A10	BUFFER FOR 2ND & 3RD BYTE OF INSTRUCTION (O/P) ('T')
0A12	O/P BUFFER FOR LOG NUMBER (FOR DATA LOGGER)
0A14 STACK ↑ 0BF1	
0BF2	ADDRESS OF NEXT SAMPLING PEROID (REPROGRAM TIMER)
0BF4	STACK POINTER BUFFER (FOR TTY MONITOR REENTRY)
0BF6	SAMPLE NUMBER (FOR CONTROLLER)
0BF8	I/P DATA & STATUS FROM TX 2
0BFA	I/P DATA & STATUS FROM TX 1
0BFC	TX 2 CURRENT CONTROL WORD
0BFD	TX 1 CURRENT CONTROL WORD
0BFE	FLAG FOR CAL'N (STARTED OR FINISHED)
0BFF	CURRENT INTERRUPT LEVEL

(B) Sub-sections:

(a) Operating System, Teletype Monitor and Data Logger:

0000	INTERRUPT DIRECTOR & SERVICE ROUTINE FOR INTR. LEVEL 0
0046	
0047	SERVICE ROUTINE FOR INTERRUPT LEVEL 1 DISPLAY OR ENTER A CONSTANT FROM THE FRONT PANEL
00DE	
00DF	SERVICE ROUTINE FOR INTERRUPT LEVEL 2 O/P A CHARACTER VIA SERIAL TRANSMISSION INTERFACE 2
013E	
013F	SERVICE ROUTINE FOR INTERRUPT LEVEL 3 I/P A CHARACTER VIA SERIAL TRANSMISSION INTERFACE 2
01C2	
01C3	SERVICE ROUTINE FOR INTERRUPT LEVEL 4 O/P A CHARACTER VIA SERIAL TRANSMISSION INTERFACE 1
0222	
0223	SERVICE ROUTINE FOR INTERRUPT LEVEL 5 I/P A CHARACTER VIA SERIAL TRANSMISSION INTERFACE 1
0288	
0289	SERVICE ROUTINE FOR INTERRUPT LEVEL 6 TIMER INTR.: SERVICE FRONT PANEL & CONTROLLER I/O
02E4	
02E5	DATA LOGGER & CONTROL CALCULATION ORGANISER
043C	
043D	SYSTEM INITIALISATION
0499	
049A	TELETYPE MONITOR
07FF	

0C00	TELETYPE MONITOR (CONTINUE)
0F02	
0F03	UTILITY SUBROUTINES
0FFF	

(c) Floating Point and Binary Arithmetic Programs:

1000	
1003	ADDITION & SUBTRACTION (FLOATING POINT)
10C0	
10C1	MULTIPLICATION (FLOATING POINT)
11AD	
11AE	DIVISION (FLOATING POINT)
1281	
1282	UTILITY SUBROUTINES (FOR FLOATING POINT ARITH.)
135D	
135E	SUBTRACTION (2-BYTE BINARY)
1364	
1365	MULTIPLICATION (2-BYTE BINARY)
139E	
139F	DIVISION (2-BYTE BINARY)
13FE	

2. Device Addresses.

(A) Output Ports:

(a) Displays on Front Panel:

Address	Device
9ØØØ	Most significant 2 digits of mantissa of constant.
88ØØ	Least significant 2 digits of mantissa of constant.
84ØØ	(D_7) M.S.B. - sign of constant ('1' = +ve) (D_6) - sign of exponent of constant ('1' = +ve) (D_5-D_o) - exponent of constant in pure binary.
82ØØ	Address of constants (in B.C.D. Ø-99).

(b) Controller Output Ports:-

81ØØ	Most significant 2 B.C.D. digits of the D/A converter (in case of analogue O/P).
8Ø8Ø	(D_7-D_4) - least significant digit of the D/A converter (in case of analogue O/P).

(c) Priority Interrupt Controller

8Ø2Ø	(D_o-D_2) current status register (D_3) SGS disable comparator and allow all interrupt to go through ('1' for enabling all interrupt).

(d) Serial Transmission Interface

8Ø1Ø	Data o/p port for TX2
8Ø11	Control register of TX2
8ØØ8	Data o/p port for TX1
8ØØ9	Control register of TX1

(e) Programmable Timer

8ØØ4	Load counter No.'Ø'
8ØØ5	Load counter No. '1'
8ØØ6	Load counter No.'2'
8ØØ7	Write Mode Word

(B) Input Ports:

(a) Switches input from the Front Panel:

Address	Device
CØØØ	Most significant byte of address on thumb switches.
AØØØ	Least significant byte of address on thumb switches.
9ØØØ	Most significant 2 digits of mantissa of constant
88ØØ	Least significant 2 digits of mantissa of constant
84ØØ	(D_7) - sign of constant ('1' = +ve) (D_6) - sign of exponent of constant ('1' = +ve) (D_5-D_o) - exponent of constant in pure binary.
82ØØ	Address of constants (in B.C.D. Ø-99)

(b) Controller Input Ports:

81ØØ	Most significant 2 B.C.D. digits of the A/D converter (in case of analogue I/P)
8Ø8Ø	(D_7-D_4) - least significant digit of the A/D converter (in case of analogue I/P). Or, (D_7-D_0) - parallel binary input (in case of digital I/P).

(c) Serial Transmission Interface:

8Ø1Ø	Data I/P port for TX2
8Ø11	Status I/P from TX2
8ØØ8	Data I/P port for TX1
8ØØ9	Status I/P from TX1.

(d) Programmable Timer

8ØØ4	Read count from counter No. 'Ø'
8ØØ5	Read count from counter No. '1'
8ØØ6	Read count from counter No. '2'.
8ØØ7	Write Mode Word.

APPENDIX 3 - TELETYPE MONITOR COMMAND DESCRIPTION

(1) The 'COPY' command - to copy a block of data from one location to another. This command is initialised by typing:

> C	starting address of source	terminating address of source	starting address of destination	CR

N.B. CR = carriage return.

(2) The 'DUMP' command - to dump data onto paper tape. This command is initialised by typing:

> D Starting address terminating address CR

(3) The 'ENTER' Command - to enter data or program into memory. This command is initialised by typing:

> E starting address CR

then the starting address will appear at the line. Data or program can be input in form of Hexadecimal. Each data when typed in would be separated by a space. Type 'RETURN' when a new line is required. The address of the next data would automatically appear on the next line where new data can be typed in.

(4) The 'GO' command - to execute the program starting from the address stated. If the X command is not initiated for presetting the registers, all the registers, all the registers and stack will automatically be set to zero. This command is initialised by typing:

> G address of program CR(or X.....)

(5) The 'LOAD' command - to load the data from paper tape. This command is initialised by typing:

> L	starting address of destination (option I)	terminate address of destination (option II)	CR

If option I and option II is not entered, data will be loaded into the memory address according to the paper tape. If option I only is entered, data in the paper tape would be loaded onto address starting from the address in option I and terminated at the end of the paper tape.

If both option I and II is entered, data in the paper tape would be loaded into address starting from address in option I and terminated when address in option II is reached.

(6) The'MODIFY' command - to modify a data in the memory. This command is initialised by typing:

> M	address	CR

The address of the data to be modified shall be displayed foll followed by the data in that address. New data can be typed in or if 'RETURN' is typed the old data will be retained. When return is typed the next address and data will be displayed. If '↑' is typed instead, the address shall be decremented instead of incremented.

(7) The 'TRACE' command - to trace error in the program. The command is initialised by typing:

> T	starting address of program	S	CR (or X.....)...		(option I)

Or:-

> T	starting address of program	terminating address of program	N	CR (or X...)		(option II)

If 'S' is typed (i.e. option I) the first instruction will be executed and all the status flags, registers, stack and stack pointer displayed. If terminating address is typed instead the status flags, registers,... .... etc. will be displayed when each instruction is executed until the suppressed until the last instruction.

The format of listing are as follows:-

NNNN II IIII AAFF BBCC DDEE HHLL SSSS PPPP

Where: 'N' s = Address of Instruction

'I' s = Instruction

'A' s = Accumulator

'F' s = Flags

'B', 'C', 'D', 'E', 'H', 'L' s = Registers

'S' s = Stack

'P' s = Stack Pointer

When the listing has been completed the execution can be continued un until another address by typing:

T	terminating address	N	CR
		(option II)	(or X .....)

The operations are the same as before.

N.B. In all cases the X command can be used for changing the registers, flags and stack.

(8) The 'WRITE' command - to display the memory content. This command is initialised by typing:

W	starting address	terminate address	CR

(9) The 'X' command - to preset or change registers, flags or stacks

in 'G' or 'T' command. This command is initialised by typing:

X ROO GG

'R' = register name it can be A (accumulator), B (register B), C (register C), D (register D), E (register E), H (register H), L (register L), P (status flags), SH (upper byte of stack), SL (lower byte of stack).

'OO' = original content of the register.

'GG' = new content of the register (to be typed in)

A '.' can be typed for exiting from 'X' command.

N.B. In all command '↑C' can ve used for exit and return to command specifying mode.

APPENDIX 4

Source of Alarm and Floating Point Number Storage Format.

(A) Source of Alarm.

The Alarm is set off by switching on the 'ALARM' light on the front panel and a horn. The source of alarm is displayed on the constant address display. The alarm can be set by outputting a hexadecimal number starting with 'A' to the Constant Address Display. It can be reset by pressing the 'ALARM' acknowledge button. The source of alarm is as follows:

Display	Source
AØ	Add./subtr. overflow (Floating point subroutine)
A1	Add./subtr. underflow. (Floating point subroutine)
A2	Multiplication overflow (" " ")
A3	Multiplication underflow (" " ")
A4	Division overflow (" " ")
A5	Division underflow (" " ")
A6	Division result infinite (" " ")
A7	3-term controller output overflow.
A8	3-term controller output underflow.
A9	Binary Multiplication overflow.
AC	Previous Control Calculation not yet finished.

(B) Floating Point Number Storage Format.

Address	Least significant 2 digit of mantissa		
Address + 1	Most significant 2 digit of mantissa		
Address + 2	M.S.B. sign of No.	sign of exp.	Exponent in Pure Binary.
	D_7	D_6	

APPENDIX 5

Summary of Subroutines

A brief summary of the subroutines are given here for quick reference. Further detail of the subroutines can be found in the listing.

Address	Name	Function
ØØ72	DSET	To display or enter a constant.
ØØC6	CDC	Convert constant address from B.C.D. to binary and true address.
Ø117	SROP2	To place a character on the output buffer of TX2.
Ø15C	SRIP2	To read a character from the input buffer of TX2.
Ø1A6	PER2	Type 'PERROR' (TX2)
Ø1FB	SROP1	To place a character on the output buffer of TX1.
Ø24B	SRIP1	To read a character from the input buffer of TX1.
Ø2E5	PROG	Performs Data Logging and arrange the control calculation.
Ø3F8	OPRU	Output Data to specific serial transmission interface (TX2 or TX1)
Ø4ØF	CRLF	Type CR & LF
Ø41A	MASK	Convert a 2 digit B.C.D. in memory whose address is in HL to ASCII and output to TX1 if first bit of reg.C is '0'. Otherwise output would be through TX2.
Ø49D	TREN	TTY monitor re entry point.
ØFØ3	INAD	Input a 4 digit Hexadecimal number from TX2 onto DE.
ØF47	SDER	Shift DE right
ØF5Ø	RASC	Input a number from TX2 onto Acc. and convert from ASCII to Hexadecimal number.
ØF6A	ECHO	Echo at character from TX2
ØF7B	OPSA	O/P HL to TX2 as 4 digit hexadecimal number.
ØF9C	INAT	I/P a 4 digit hexadecimal number from TX2 onto DE and wait for a CR.
ØFB1	CASC	Convert a one digit hexadecimal number to ASCII and O/P through TX2.
ØFC3	DRD	Disable paper tape reader at TX2.
ØFDC	TYSP	O/P a 'space' to TX2.
ØFE4	TECH	Repeatedly type 18 characters (to be specified).
ØFF9	CR	O/P CR L LF to TX2
1ØØ3	SUMM	Floating Point Addition and Subtraction.
1ØC1	MUPY	Floating Point Multiplication.
11AE	DVD	Floating Point Division
1282	ADDSR	B.C.D. addition of BC and DE with result in DE.
129B	SUBSR	B.C.D. subtraction of BC and DE (DE-BC) with result in DE.
12CØ	ROPD	Round off a B.C.D. number in DE if CY is set. (i.e. rotate CY into DE and round off).

Address	Name	Function
12Ø6	RBCR	Rotate BC right
12ED	RDER	Rotate DE right
12F4	RDEL	Rotate DE left
12FB	RHLL	Rotate HL left
13Ø2	RBCF	Rotate BC right 4 times.
135E	BSU	Binary subtraction (HL=HL-DE)
1365	BMU	Binary multiplication (HL=HL*DE)
139F	BDV	Binary Division (HL=HL/DE)

APPENDIX 6

Listing of System Softwares in the "Algorithm Development/ Implementation System".

(A) The Operating System, the Teletype Monitor and the Data Logger.

(B) The Floating Point Arithematic Program.

(C) The Double Byte Binary Arithematic Program.

```
                .TITLE "OPERATING SYSTEM - 8080"
                .HEX
                USER=1400
                INTER=1000
                ;INTERRUPT LEVEL 7
                ;RESET AND INITIALISATION
0000    F3              DI
0001    31              LXI  SP,0BF1
0002    F1
0003    0B
0004    C3              JMP INIT
0005    3D
0006    04
0007    00              NOP
                ;INTERRUPT LEVEL 6
                ;TIMER INTERRUPT
0008    F5              PUSH PSW
0009    E5              PUSH H
000A    21              LXI  H,0BFF  ;LOAD ADDR. OF INTR. REG.
000B    FF
000C    0B
000D    C3              JMP  IN2
000E    89
000F    02
                ;INTERRUPT LEVEL 5
                ;SERIAL INPUT 1 READY
0010    F5              PUSH PSW
0011    E5              PUSH H
0012    21              LXI  H,0BFF
0013    FF
0014    0B
0015    C3              JMP  IN3
0016    23
0017    02
                ;INTERRUPT LEVEL 4
                ;SERIAL OUTPUT 1 READY
0018    F5              PUSH PSW
0019    E5              PUSH H
001A    21              LXI  H,0BFF
001B    FF
001C    0B
001D    C3              JMP  IN4
001E    C3
001F    01
                ;INTERRUPT LEVEL 3
                ;SERIAL INPUT 2 READY
0020    F5              PUSH PSW
0021    E5              PUSH H
0022    21              LXI  H,0BFF
0023    FF
0024    0B
0025    C3              JMP  IN5
0026    3F
0027    01
                ;INTERRUPT LEVEL 2
                ;SERIAL OUTPUT 2 READY
0028    F5              PUSH PSW
0029    E5              PUSH H
002A    21              LXI  H,0BFF
002B    FF
002C    0B
002D    C3              JMP  IN6
002E    DF
002F    00
                ;INTERRUPT LEVEL 1
                ;DISPLAY / ENTER CONSTANTS
0030    F5              PUSH PSW
0031    E5              PUSH H
0032    21              LXI  H,0BFF
0033    FF
0034    0B
0035    C3              JMP  IN7
0036    47
0037    00
                ;INTERRUPT LEVEL 0
                ;SERVICE THE HARDWARE INTERRUPT FROM EXECUTE SWITCH
0038    3A              LDA  0C000   ;READ M. S. BYTE
0039    00
003A    C0
003B    67              MOV  H,A     ;           OF ADDRESS
003C    3A              LDA  0A000   ;READ L. S. BYTE
003D    00
003E    A0
003F    6F              MOV  L,A     ;           OF ADDRESS
0040    3E              MVI  A,0F    ;REPROGRAM PIORITY
0041    0F
0042    32              STA  8020    ;     INTERRUPT CONTROLLER
0043    20
0044    80
0045    FB              EI
0046    E9              PCHL         ;JMP TO ADDR. STORED IN HL
```

```
                        ;*************************************************
                        ;
                        ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 1
                        ;       TO DISPLAY OR ENTER A CONSTANT
                        ;            FROM FRONT PANEL
                        ;
                        ;*************************************************
0047    D5      IN7:    PUSH D
0048    C5              PUSH B
0049    7E              MOV  A,M        ;READ PREVIOUS INTR. LEVEL
004A    F5              PUSH PSW        ;SAVE IT ON STACK
004B    3E              MVI  A,06       ;REPROGRAM PIORITY
004C    06
004D    77              MOV  M,A        ;       INTERRUPT
004E    32              STA  8020       ;       CONTROLLER
004F    20
0050    80
0051    FB              EI
0052    CD              CALL DSET
0053    72
0054    00
                        ;THIS SECTION RESTORES ACC., REG. & INTR. REG.
                        ;ON RETURN FROM INTERRUPT ROUTINE.
0055    F1      INTRE:  POP  PSW        ;FETCH PREVIOUS INTR. LEVEL
0056    F3              DI
                        ;TO TEST IF INTR. LEVEL RETURNED TO 0. THEN ALLOW
                        ;CONTROL CALCULATION TO CONTINUE.
0057    47      TST:    MOV  B,A        ;STORE ACC. IN B
0058    FE              CPI  07
0059    07
005A    DA              JC   NST        ;IF INTR. LEVEL>0
005B    65
005C    00
005D    3A              LDA  0BFE       ;LOAD START OR FINISH STATUS
005E    FE
005F    0B
0060    E6              ANI  0F0        ;IF START CALCULATION
0061    F0
0062    C4              CNZ  PROG       ;START CALCULATION
0063    E5
0064    02
0065    78      NST:    MOV  A,B        ;RESTORE ACC.
0066    32              STA  0BFF       ;RESET INTR. REGISTER
0067    FF
0068    0B
0069    32              STA  8020       ;REPROGRAM PIORITY INTR. CONTROLLER
006A    20
006B    80
006C    C1              POP  B
006D    D1              POP  D
006E    E1              POP  H
006F    F1              POP  PSW
0070    FB              EI
0071    C9              RET
                        ;-------------------------------------------------
                        ;SUBROUTINE DSET
                        ;       FUNCTION: TO DISPLAY/ENTER A CONSTANT
                        ;       INPUTS  : CONSTANT TABLE, SP
                        ;       OUTPUTS : CONSTANT TABLE, SP
                        ;       CALLS   : CDC
                        ;       DESTROYS: A,B,C,D,E,H,L
                        ;-------------------------------------------------
0072    21      DSET:   LXI  H,8200
0073    00
0074    82
0075    7E              MOV  A,M        ;I/P ADDR. OF CONST.
0076    77              MOV  M,A        ;O/P ADDR. OF CONST.
0077    CD              CALL CDC        ;CONDITION CONST. ADDR.
0078    C6
0079    00
007A    06              MVI  B,00       ;CLEAR B
007B    00
007C    11              LXI  D,8400     ;LOAD ADDR. OF EXPONENTIAL
007D    00
007E    84
007F    3A              LDA  8041       ;CHECK
0080    41
0081    80
0082    E6              ANI  10         ;    ENTER/DISPLAY
0083    10
0084    C2              JNZ  DISP
0085    B9
0086    00
                        ;START TO ENTER CONSTANTS
0087    1A      ENT:    LDAX D          ;I/P CONST.
0088    12              STAX D          ;O/P CONST.
0089    77              MOV  M,A        ;STORE CONST. IN TABLE
                        ;INCREMENT TABLE POINTER
008A    04      INC:    INR  B          ;INCREMENT B (BYTE COUNTER)
008B    3E              MVI  A,01
008C    01
```

```
                B8          CMP  B        ;IF B=1
                C2          JNZ  BTWO     ;IF B NOT=1
                97
                00
                11          LXI  D,9000   ;LOAD ADDR. OF U. DECIMAL
                00
                90
                C3          JMP  IMD
                9F
                00
                            ;PREPARE FOR LOADING OF LOWER DECIMAL DIGITS
                3C BTWO:    INR  A
                B8          CMP  B
                C2          JNZ  PEND     ;IF B NOT=2
                A3
                00
                11          LXI  D,8800   ;LOAD ADDR. OF L. DEC.
                00
                88
                2B IMD:     DCX  H        ;DECREMENT HL
                C3          JMP  ENT
                B7
                00
                            ;THIS PART ENDS A TIMER INTERRUPT ROUTINE
                7C PEND:    MOV  A,H      ;TEST WHETHER
                FE          CPI  08       ;     SAMPLING
                08
                C0          RNZ           ;     RATE
                7D          MOV  A,L      ;     HAD
                FE          CPI  00       ;     CHANGED
                00
                C0          RNZ           ;IF HL=0800 CONTINUE
                3E LDT:     MVI  A,37
                37
                32          STA  8007     ;LOAD CONTROL WORD TO COUNTER 0
                07
                80
                11          LXI  D,8004
                04
                80
                7E          MOV  A,M
                12          STAX D        ;LOAD L.S.B. OF TIMER
                23          INX  H
                7E          MOV  A,M
                12          STAX D        ;LOAD M.S.B. OF TIMER
                C9          RET
                            ;THIS SECTION DISPLAY THE CONSTANT ON FRONT PANEL
                7E DISP:    MOV  A,M      ;FETCH EXP.
                12          STAX D        ;O/P EXP.
                2B          DCX  H
                7E          MOV  A,M      ;FETCH M.S. DIGIT
                32          STA  9000     ;O/P M.S. DIGIT
                00
                90
                2B          DCX  H
                7E          MOV  A,M      ;FETCH L.S. DIGIT
                32          STA  8800     ;O/P L.S. DIGIT
                00
                88
                C9          RET
                            ;-----------------------------------------------
                            ;SUBROUTINE CDC
                            ;       FUNCTION: CONVERT CONSTANT ADDR. FROM B.C.D.
                            ;                 TO BINARY AND TRUE ADDR.
                            ;       INPUTS  : A
                            ;       OUTPUTS : H,L
                            ;       CALLS   : NONE
                            ;       DESTROYS: A,B,C,H,L
                            ;       DESCRIPTION: THIS SUBROUTINE CONVERTS THE
                            ;                    CONSTANT ADDR. TO CORRESPONDING
                            ;                    ADDR. ON THE CONSTANT TABLE IN
                            ;                    MEMORY.
                            ;-----------------------------------------------
                47 CDC:     MOV  B,A      ;START B.C.D. TO BINARY
                E6          ANI  0F       ;               CONVERSION
                0F
                4F          MOV  C,A      ;STORE LOWER B.C.D.
                78          MOV  A,B
                E6          ANI  0F0      ;EXTRACT UPPER B.C.D.
                F0
                0F          RRC           ;MULTIPLY
                47          MOV  B,A      ;     UPPER
                0F          RRC           ;     B.C.D. DIGIT
                0F          RRC           ;     BY
                80          ADD  B        ;     TEN
                81          ADD  C        ;ADD PRODUCT TO LOWER B.C.D.
                            ;END BCD - BINARY CONVERSION AND START CONVERTING
                            ;TO ADDRESS IN TABLE
                47 CTA:     MOV  B,A
                80          ADD  B
                80          ADD  B        ;MULTIPLY BY 3
                6F          MOV  L,A
```

```
129 00D7    3E          MVI  A,00
    00D8    00
130 00D9    CE          ACI  00       ;ADD 2048D
    00DA    00
131 00DB    67          MOV  H,A      ;TRUE ADDR. IN HL
132 00DC    23          INX  H        ;INCREMENT H
133 00DD    23          INX  H        ;       BY 2
134 00DE    C9          RET
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE    3

  1                 ;***********************************************
  2                 ;
  3                 ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 2
  4                 ;        TO OUTPUT A CHARACTER VIA SERIAL
  5                 ;        TRANSMISSION INTERFACE 2 (TX 2)
  6                 ;
  7                 ;***********************************************
  8                 ;SECTION FOR SAVING ALL STATUS AFTER INTERRUPT
  9 00DF    D5  IN6:    PUSH D
 10 00E0    C5          PUSH B
 11 00E1    7E          MOV  A,M      ;READ PREVIOUS INTR. LEVEL
 12 00E2    F5          PUSH PSW      ;SAVE IT ON STACK
 13 00E3    3E          MVI  A,05     ;REPROGRAM PIORITY
    00E4    05
 14 00E5    77          MOV  M,A      ;          INTERRUPT
 15 00E6    32          STA  8020     ;          CONTROLLER
    00E7    20
    00E8    80
 16                 ;SECTION FOR ORGANISING BUFFER POINTER AND TRANSMISSION
 17 00E9    2A          LHLD 0990     ;LOAD HL WITH BUFFER PTER.
    00EA    90
    00EB    09
 18 00EC    7D          MOV  A,L      ;TEST
 19 00ED    EE          XRI  92       ;  IF BUFFER PTER.=0992
    00EE    92
 20 00EF    C2          JNZ  OUDA2
    00F0    00
    00F1    01
 21                 ;SECTION FOR DISABLING TRANSMISSION
 22 00F2    3A          LDA  0BFC     ;READ TX2 CONTROL WORD
    00F3    FC
    00F4    0B
 23 00F5    E6          ANI  0DE      ;CLEAR TX RDY. FLAG
    00F6    DE
 24 00F7    32          STA  0BFC     ;STORE TX2 CONTROL WORD
    00F8    FC
    00F9    0B
 25 00FA    32          STA  8011     ;PROGRAM TX2
    00FB    11
    00FC    80
 26 00FD    C3          JMP  INTRE    ;TRANSMISSION DISABLED/END ROUTINE
    00FE    55
    00FF    00
 27                 ;SECTION FOR OUTPUTING DATA FROM BUFFER TO TX2
 28 0100    3A  OUDA2:  LDA  0992     ;FETCH DATA FROM BOTTOM
    0101    92
    0102    09
 29 0103    32          STA  8010     ;O/P DATA TO TX2
    0104    10
    0105    80
 30 0106    2B          DCX  H        ;DECREMENT HL
 31 0107    22          SHLD 0990     ;STORE BUFFER PTER.
    0108    90
    0109    09
 32 010A    46  LOOPS2: MOV  B,M      ;B_BUFFER
 33 010B    71          MOV  M,C      ;BUFFER_C
 34 010C    48          MOV  C,B      ;C_B
 35 010D    2B          DCX  H        ;DECREMENT HL
 36 010E    3E          MVI  A,91
    010F    91
 37 0110    95          SUB  L        ;IF HL<START ADDR.
 38 0111    FA          JM   LOOPS2   ;IF NOT
    0112    0A
    0113    01
 39 0114    C3          JMP  INTRE    ;END INTR. SERVICE ROUTINE
    0115    55
    0116    00
 40                 ;***********************************************
 41                 ;       OUTPUT SUBROUTINE FOR OUTPUTTING TO TX2
 42                 ;       FUNCTION: TO PLACE A OUTPUT CHARACTER ON
 43                 ;                 THE OUTPUT BUFFER OF TX2
 44                 ;       INPUTS  : C
 45                 ;       OUTPUTS : NONE
 46                 ;       DISTROYS: NOTHING
 47                 ;       DESCRIPTION: TO ENTER THIS ROUTINE, FIRST
 48                 ;                    PUT OUTPUT CHARACTER ON REG.
 49                 ;                    C, THEN CALL SROP2.
 50                 ;***********************************************
 51 0117    F5  SROP2:  PUSH PSW
 52 0118    E5          PUSH H
 53 0119    2A  WAIT2:  LHLD 0990     ;FETCH BUFFER PTER.
    011A    90
    011B    09
```

```
 54 011C    3E          MVI  A,0F2   ;TEST
    011D    F2
 55 011E    95          SUB  L       ;IF BUFFER PTER.>LIMIT
 56 011F    FA          JM   WAIT2
    0120    19
    0121    01
 57 0122    F3          DI
 58 0123    71          MOV  M,C     ;BUFFER_C
 59 0124    23          INX  H       ;INCREMENT HL
 60 0125    22          SHLD 0990    ;STORE BUFFER PTER.
    0126    90
    0127    09
 61                ;SECTION FOR ENABLING TRANSMISSION
 62 0128    3A          LDA  0BFC    ;FETCH TX2 CONTROL WORD
    0129    FC
    012A    0B
 63 012B    E6          ANI  01
    012C    01
 64 012D    C2          JNZ  ENAB2   ;IF TX2 ENABLED
    012E    3B
    012F    01
 65 0130    3A          LDA  0BFC    ;FETCH TX2 CONTROL WORD
    0131    FC
    0132    0B
 66 0133    F6          ORI  21      ;SET TX RDY. FLAG
    0134    21
 67 0135    32          STA  0BFC    ;STORE TX2 CONTROL WORD
    0136    FC
    0137    0B
 68 0138    32          STA  8011    ;PROGRAM TX2
    0139    11
    013A    80
 69 013B    FB ENAB2:   EI
 70 013C    E1          POP  H       ;END OF SUBROUTINE
 71 013D    F1          POP  PSW
 72 013E    C9          RET
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE    4

  1                ;*********************************************
  2                ;
  3                ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 3
  4                ;        TO INPUT A CHARACTER VIA SERIAL
  5                ;        TRAMISSION INTERFACE 2 (RX2)
  6                ;
  7                ;*********************************************
  8                ;SECTION FOR SAVING ALL STATUS AFTER INTR.
  9 013F    D5 IN5:     PUSH D
 10 0140    C5          PUSH B
 11 0141    7E          MOV  A,M
 12 0142    F5          PUSH PSW
 13 0143    3E          MVI  A,04
    0144    04
 14 0145    77          MOV  M,A
 15 0146    32          STA  8020
    0147    20
    0148    80
 16                ;THIS SECTION READS THE DATA AND ORGANIZE ALL
 17                ;THE FLAGS
 18 0149    21          LXI  H,8011  ;LOAD ADDR. OF TX2 STATUS REG.
    014A    11
    014B    80
 19 014C    11          LXI  D,0BF9  ;LOAD ADDR. OF RX2 STATUS BUF.
    014D    F9
    014E    0B
 20 014F    7E          MOV  A,M     ;READ TX2 STATUS WORD.
 21 0150    EB          XCHG
 22 0151    77          MOV  M,A     ;STORE STATUS IN BUFFER
 23                ;INPUT DATA
 24 0152    EB          XCHG
 25 0153    2B          DCX  H       ;SET INPUT PORT ADDR.
 26 0154    1B          DCX  D       ;SET INPUT BUFFER ADDR.
 27 0155    7E          MOV  A,M     ;INPUT DATA
 28 0156    FB          EI
 29 0157    EB          XCHG
 30 0158    77          MOV  M,A     ;STORE DATA IN BUFFER
 31 0159    C3          JMP  INTRE   ;END INTR. SERVICE ROUTINE
    015A    55
    015B    00
 32                ;*********************************************
 33                ;       INPUT SUBROUTINE FOR INPUTTING FROM TX2
 34                ;       FUNCTION: TO READ A CHARACTER FROM THE
 35                ;                 INPUT BUFFER
 36                ;       INPUTS  : NONE
 37                ;       OUTPUTS : ACC, RESET I/P BUFFER & STATUS.
 38                ;       CALLS   : SKOP2
 39                ;       DISTROYS: NOTHING
 40                ;       DISCRIPTIONS: THIS ROUTINE PUTS A 7 BIT
 41                ;                     DATA ON THE ACCUMULATOR WITH
 42                ;                     1ST BIT (8TH BIT) EQUAL TO
 43                ;                     '0'.
 44                ;                     THIS ROUTINE WOULD ALSO CHECK
 45                ;                     FLAMING ERROR AND PARITY
 46                ;                     ERROR
```

```
47                      ;                       'FERROR' = FLAMING ERROR
48                      ;                       'PERROR' = PARITY ERROR
49                      ;                       IF PARITY ERROR - 1ST BIT
50                      ;                       (M.S.B.) OF DATA WOULD BE
51                      ;                       SET TO '1'.
52                      ;*********************************************
53 015C  E5 SRIP2:  PUSH H
54 015D  C5         PUSH B
55 015E  2A RDY2:   LHLD 0BF8     ;FETCH DATA & STATUS
   015F  F8
   0160  0B
56 0161  7C         MOV  A,H
57 0162  E6         ANI  2        ;TEST IF RX RDY IS SET
   0163  02
58 0164  CA         JZ   RDY2     ;IF READY
   0165  5E
   0166  01
59 0167  AF CLR2:   XRA  A
60 0168  32         STA  0BF9     ;CLEAR STATUS BUFFER
   0169  F9
   016A  0B
61                  ;ERROR CHECK
62 016B  7D         MOV  A,L      ;CLEAR
63 016C  E6         ANI  7F       ;     PARITY
   016D  7F
64 016E  6F         MOV  L,A      ;     BIT
65 016F  7C         MOV  A,H      ;ACC. _ STATUS
66 0170  E6         ANI  8        ;TEST IF PARITY ERROR
   0171  08
67 0172  CA         JZ   FER2     ;IF NOT
   0173  7C
   0174  01
68 0175  7D         MOV  A,L      ;SET 1ST BIT
69 0176  F6         ORI  80       ;     OF DATA '1'
   0177  80
70 0178  6F         MOV  L,A      ;     IF PARITY ERROR
71 0179  CD         CALL PER2     ;TYPE 'PERROR'
   017A  A6
   017B  01
72 017C  7C FER2:   MOV  A,H
73 017D  E6         ANI  20       ;IF FRAMING ERROR
   017E  20
74 017F  CA         JZ   CER2     ;IF NOT
   0180  8D
   0181  01
75 0182  0E         MVI  C,'F'    ;TYPE
   0183  46
76 0184  CD         CALL SROP2    ;     'FERROR'
   0185  17
   0186  01
77 0187  CD         CALL ER2
   0188  AB
   0189  01
78 018A  C3         JMP  YFR2
   018B  93
   018C  01
79 018D  7D CER2:   MOV  A,L
80 018E  E6         ANI  80
   018F  80
81 0190  CA         JZ   EIP2     ;IF NO PERROR
   0191  9D
   0192  01
82                  ;CLEAR ERROR FLAGS
83 0193  F3 YFR2:   DI
84 0194  3A         LDA  0BFC     ;FETCH CONTROL WORD
   0195  FC
   0196  0B
85 0197  F6         ORI  10
   0198  10
86 0199  32         STA  8011     ;ERROR FLAGS CLEARED
   019A  11
   019B  80
87 019C  FB         EI
88                  ;SECTION FOR ENDING SUBROUTINE
89 019D  7D EIP2:   MOV  A,L      ;RESTORE DATA TO ACC.
90 019E  C1         POP  B
91 019F  E1         POP  H
92 01A0  FE         CPI  03       ;IF ETX (^C) GO TO REENTRY
   01A1  03
93 01A2  CA         JZ   TREN     ;     OF TTY MONITOR
   01A3  9D
   01A4  04
94 01A5  C9         RET
95                  ;---------------------------------------------
96                  ;SUBROUTINE FOR TYPING 'PERROR'
97                  ;---------------------------------------------
98 01A6  0E PER2:   MVI  C,'P'
   01A7  50
99 01A8  CD         CALL SROP2
   01A9  17
   01AA  01
```

```
100                 ;SUBROUTINE FOR TYPING 'ERROR'
101 01AB    0E ER2:     MVI  C,'E'
    01AC    45
102 01AD    CD          CALL SROP2
    01AE    17
    01AF    01
103 01B0    0E          MVI  C,'R'
    01B1    52
104 01B2    CD          CALL SROP2
    01B3    17
    01B4    01
105 01B5    CD          CALL SROP2
    01B6    17
    01B7    01
106 01B8    0E          MVI  C,'O'
    01B9    4F
107 01BA    CD          CALL SROP2
    01BB    17
    01BC    01
108 01BD    0E          MVI  C,'R'
    01BE    52
109 01BF    CD          CALL SROP2
    01C0    17
    01C1    01
110 01C2    C9          RET
TELETYPE MONITOR - 8000              CALOS-80 V2.12 00/00/70 PAGE    5

  1                 ;*********************************************
  2                 ;
  3                 ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 4
  4                 ;        TO OUTPUT A CHARACTER VIA SERIAL
  5                 ;        TRANSMISSION INTERFACE 1 (TX1)
  6                 ;
  7                 ;*********************************************
  8                 ;SECTION FOR SAVING ALL STATUS AFTER INTERRUPT
  9 01C3    D5  IN4:    PUSH D
 10 01C4    C5          PUSH B
 11 01C5    7E          MOV  A,M      ;READ PREVIOUS INTR. LEVEL
 12 01C6    F5          PUSH PSW      ;SAVE IT ON STACK
 13 01C7    3E          MVI  A,03     ;REPROGRAN PRORITY
    01C8    03
 14 01C9    77          MOV  M,A      ;           INTERRUPT
 15 01CA    32          STA  8020     ;           CONTROLLER
    01CB    20
    01CC    80
 16                 ;SECTION FOR ORGANISING BUFFER POINTER AND
 17 01CD    2A          LHLD 092C     ;LOAD HL WITH BUFFER PTER.
    01CE    2C
    01CF    09
 18 01D0    7D          MOV  A,L      ;TEST
 19 01D1    EE          XRI  2E       ;    IF BUFFER PTER. = 0930
    01D2    2E
 20 01D3    C2          JNZ  OUDA1
    01D4    E4
    01D5    01
 21                 ;SECTION FOR DISABLING TRANSMISSION
 22 01D6    3A          LDA  0BFD     ;READ TX2 CONTROL WORD
    01D7    FD
    01D8    0B
 23 01D9    E6          ANI  0DE      ;CLEAR TX RDY. FLAG
    01DA    DE
 24 01DB    32          STA  0BFD     ;STORE TX2 CONTROL WORD
    01DC    FD
    01DD    0B
 25 01DE    32          STA  8009     ;PROGRAM TX2
    01DF    09
    01E0    80
 26 01E1    C3          JMP  INTRE    ;TRANSMISSION DISABLED/END ROUTINE
    01E2    55
    01E3    00
 27                 ;SECTION FOR OUTPUTING DATA FROM BUFFER TO TX2
 28 01E4    3A  OUDA1:  LDA  092E     ;FETCH DATA FROM BOTTOM
    01E5    2E
    01E6    09
 29 01E7    32          STA  8008     ;O/P DATA TO TRANSMITTER
    01E8    08
    01E9    80
 30 01EA    2B          DCX  H        ;DECREMENT H & L
 31 01EB    22          SHLD 092C     ;STORE BUFFER PTER.
    01EC    2C
    01ED    09
 32 01EE    46  LOOPS1: MOV  B,M      ;B_BUFFER
 33 01EF    71          MOV  M,C      ;BUFFER _C
 34 01F0    48          MOV  C,B      ;C_B
 35 01F1    2B          DCX  H        ;DECREMENT HL
 36 01F2    3E          MVI  A,2D
    01F3    2D
 37 01F4    95          SUB  L        ;IF HL<START ADDR.
 38 01F5    FA          JM   LOOPS1   ;IF NOT
    01F6    EE
    01F7    01
 39 01F8    C3          JMP  INTRE    ;END INTR. SERVICE ROUTINE
    01F9    55
    01FA    00
```

```
 88                     ;SECTION FOR ENDING SUBROUTINE
 89 0285    7D EIP1:    MOV  A,L      ;RESTORE DATA TO ACC.
 90 0286    C1          POP  B
 91 0287    E1          POP  H
 92 0288    C9          RET
TELETYPE MONITOR - 8080            CALOS-80 V2.12 00/00/70 PAGE    7

  1                     ;***********************************************
  2                     ;
  3                     ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 6
  4                     ;        TO SERVE THE TIMER INTERRUPT, SCAN
  5                     ;        THE FRONT PANEL AND START CONTROL
  6                     ;        CALCULATION
  7                     ;
  8                     ;***********************************************
  9                     ;SECTION FOR SAVING ALL STATUS AFTER INTR.
 10 0289    D5 IN2:     PUSH D
 11 028A    C5          PUSH B
 12 028B    7E          MOV  A,M
 13 028C    F5          PUSH PSW
 14 028D    3E          MVI  A,01
    028E    01
 15 028F    77          MOV  M,A
 16 0290    32          STA  8020
    0291    20
    0292    80
 17                     ;REPROGRAM TIMER
 18 0293    2A          LHLD 0BF2
    0294    F2
    0295    0B
 19 0296    7E          MOV  A,M      ;FETCH NEW ADDR.
 20 0297    E6          ANI  3F
    0298    3F
 21 0299    EB          XCHG
 22 029A    CD          CALL CTA      ;CONV. NEW ADDR. TO TRUE ADDR.
    029B    D3
    029C    00
 23 029D    7A          MOV  A,D      ;IF
 24 029E    BC          CMP  H        ;  NEW
 25 029F    C2          JNZ  NWR      ;  ADDR.
    02A0    A7
    02A1    02
 26 02A2    7B          MOV  A,E      ;  =
 27 02A3    BD          CMP  L        ;  OLD
 28 02A4    CA          JZ   ODR      ;  ADDR.
    02A5    AF
    02A6    02
 29 02A7    22 NWR:     SHLD 0BF2     ;STORE NEW ADDR.
    02A8    F2
    02A9    0B
 30 02AA    2B          DCX  H
 31 02AB    2B          DCX  H
 32 02AC    CD          CALL LDT      ;LOAD NEW COUNT TO COUNTER
    02AD    AB
    02AE    00
 33                     ;SECTION FOR O/P TO D/A
 34 02AF    FB ODR:     EI
 35 02B0    2A          LHLD 0803     ;FETCH DATA
    02B1    03
    02B2    08
 36 02B3    7C          MOV  A,H
 37 02B4    32          STA  8100     ;O/P M.S.B.S. TO D/A
    02B5    00
    02B6    81
 38 02B7    7D          MOV  A,L
 39 02B8    32          STA  8080     ;O/P L.S.B. TO D/A
    02B9    80
    02BA    80
 40                     ;SECTION FOR WAITING A/D CONVERSION TO COMPLETE
 41 02BB    3A CONV:    LDA  8041
    02BC    41
    02BD    80
 42 02BE    E6          ANI  80
    02BF    80
 43 02C0    CA          JZ   CONV
    02C1    BB
    02C2    02
 44                     ;SECTION FOR I/P DATA FROM A/D OR OTHER PORTS.
 45 02C3    21          LXI  H,0807   ;LOAD DISTINATION ADDR.
    02C4    07
    02C5    08
 46 02C6    3A          LDA  8100     ;I/P M.S.B. FROM A/D
    02C7    00
    02C8    81
 47 02C9    77          MOV  M,A      ;STORE DATA
 48 02CA    2B          DCX  H
 49 02CB    3A          LDA  8080     ;I/P L.S.B. FROM A/D
    02CC    80
    02CD    80
 50 02CE    77          MOV  M,A      :STORE DATA
```

```
 51                ;SECTION FOR TESTING WHETHER PREVIOUS CALCULATION
 52                ;BEEN COMPLETED.
 53 02CF  21'              LXI  H,0BFE  ;FETCH START/FINISH
    02D0  FE
    02D1  0B
 54 02D2  7E               MOV  A,M     ;      STATUS WORD.
 55 02D3  E6               ANI  0F      ;TEST
    02D4  0F
 56 02D5  C2               JNZ  STSF    ;      IF FINISHED
    02D6  E0
    02D7  02
 57 02D8  3E               MVI  A,0AC   ;IF NOT COMPLETED
    02D9  AC
 58 02DA  32               STA  8200    ;         SET ALARM
    02DB  00
    02DC  82
 59 02DD  C3               JMP  INTRE
    02DE  55
    02DF  00
 60 02E0  2F STSF:         CMA          ;SET START & RESET FINISH BITS
 61 02E1  77               MOV  M,A     ;STORE S/F STATUS
 62 02E2  C3               JMP  INTRE
    02E3  55
    02E4  00
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE    8

  1                ;***********************************************
  2                ;       DATA LOGGING & CONTROL CALCULATION SUBROUTINE
  3                ;       FUNCTION: THIS PROGRAM PERFORMS DATA LOGGING
  4                ;                 AND ARRANGE THE CONTROL CALCULATION.
  5                ;       INPUTS  : NONE
  6                ;       OUTPUTS : NONE
  7                ;       DESTROYS: NONE
  8                ;       DESCRIPTION: THIS SUBROUTINE CAN BE DIVIDED
  9                ;                    INTO TWO PARTS:
 10                ;                    (1) DATA LOGGING
 11                ;                        ACCEPTS PROGRAMMING FROM THE
 12                ;                        LAST TEN CONSTANTS (90-99).
 13                ;                    (2) MAINTAIN THE CONTROL CALN.
 14                ;***********************************************
 15                ;SECTION FOR RESETTING THE STATUS WORDS & INTR. LEVEL
 16 02E5  C5 PROG:         PUSH B
 17 02E6  3E               MVI  A,07    ;SET
    02E7  07
 18 02E8  32               STA  0BFF    ;    INTR.
    02E9  FF
    02EA  0B
 19 02EB  32               STA  8020    ;          LEVEL
    02EC  20
    02ED  80
 20 02EE  3E               MVI  A,0     ;RESET
    02EF  00
 21 02F0  32               STA  0BFE    ;      START CALN
    02F1  FE
    02F2  0B
 22 02F3  FB               EI
 23                ;SECTION FOR DATA LOGGING
 24 02F4  3A               LDA  8041    ;FETCH STATUS WORD
    02F5  41
    02F6  80
 25 02F7  E6               ANI  20      ;TEST IF DATA LOGGER ON
    02F8  20
 26 02F9  C2               JNZ  SDPY    ;IF OFF
    02FA  D1
    02FB  03
 27 02FC  21               LXI  H,092B  ;*START DATA LOGGING*
    02FD  2B
    02FE  09
 28 02FF  3E               MVI  A,0FF   ;SET ACC.
    0300  FF
 29 0301  A6               ANA  M       ;IF NUMBER OF SAMPLES
 30 0302  CA               JZ   TNLG    ;   BEFORE LOGGING(L1)=0
    0303  09
    0304  03
 31 0305  35               DCR  M       ;DECREMENT L1
 32 0306  C3               JMP  SDPY    ;NOT LOGGING
    0307  D1
    0308  03
 33 0309  2A TNLG:         LHLD 0929    ;FETCH NO. OF DATA TO BE LOG.(L2)
    030A  29
    030B  09
 34 030C  3E               MVI  A,0     ;CLEAR ACC.
    030D  00
 35 030E  BC               CMP  H       ;IF L2=0
 36 030F  C2               JNZ  DECN    ;      I.E. LOGGING
    0310  16
    0311  03
 37 0312  BD               CMP  L       ;      HAS
 38 0313  CA               JZ   SDPY    ;      FINISHED
    0314  D1
    0315  03
```

```
                7D DECN:    MOV  A,L      ;DECREMENT
                C6          ADI  99       ;
                99
                27          DAA           ;
                6F          MOV  L,A      ;
                7C          MOV  A,H      ;
                CE          ACI  99       ;
                99
                27          DAA           ;
                67          MOV  H,A      ;              L2
                22          SHLD 0929     ;STORE L2
                29
                09
                3A          LDA  0927     ;FETCH L4
                27
                09
                57          MOV  D,A      ;       ONTO D
                E6          ANI  0F0      ;TEST O/P DEVICE
                F0
                C2          JNZ  STX2     ;SET O/P DEVICE TO TX2
                31
                03
                0E          MVI  C,0      ;CLEAR C (SET O/P TO TX1)
                00
                C3          JMP  STX1
                33
                03
                0E STX2:    MVI  C,80     ;SET C (1ST BIT)
                80
                   ;THIS SECTION TYPES THE LOG NUMBER
                22 STX1:    SHLD 0A12     ;SAVE LOG NO. IN BUF.
                12
                0A
                21          LXI  H,0A13
                13
                0A
                CD          CALL MASK     ;O/P M.S. 2 DIGIT
                1A
                04
                2B          DCX  H
                CD          CALL MASK     ;O/P L.S. 2 DIGIT
                1A
                04
                2A          LHLD 0A12     ;RESTORE LOG NO.
                12
                0A
                3E          MVI  A,' '    ;TYPE SPACE
                20
                CD          CALL OPRU
                F8
                03
                   ;RESET NUMBER OF SAMPLES BETWEEN LOG.
                3A          LDA  0928     ;PUT L3
                28
                09
                32          STA  092B     ;       IN L1
                2B
                09
                21          LXI  H,0926   ;PUT ADDR. OF 1ST.
                26
                09
                E5          PUSH H        ;    CONSTANT ON STACK
                1E          MVI  E,0      ;SET CONST. ADDR. LOCATION
                00                        ;TO LOWER DIGIT
                   ;START TYPING DATA
                3E STLG:    MVI  A,0F     ;IF LOWER L4 (NO. OF
                0F
                A2          ANA  D        ;    CONSTANT)=0
                C2          JNZ  LOG      ;IF NOT EQUAL TO 0 TYPE NEXT DATA
                61
                03
                E1          POP  H        ;RESTORE ADDR. OF 1ST. CONST.
                CD          CALL CRLF     ;TYPE CR & LF
                0F
                04
                C3          JMP  SDPY     ;TERMINATE LOGGING & START DISP.
                D1
                03
                   ;TYPE DATA
                3A LOG:     LDA  0927     ;TYPE CR & LF
                27
                09
                92          SUB  D        ;     AFTER
                FE          CPI  6        ;     6 DATA
                06
                CC          CZ   CRLF     ;     BEEN TYPED
                0F
                04
                E1          POP  H        ;RESTORE ADDR. OF CONST. ADDR.
                46          MOV  B,M      ;FETCH CONST. ADDR.
                2B          DCX  H        ;DEC. ADDR. OF CONST. ADDR.
```

```
 86 036D   3E          MVI  A,0FF   ;TEST AND COMPLEMENT
    036E   FF
 87 036F   AB          XRA  E       ;     REG. INDICATING CONST.
 88 0370   5F          MOV  E,A     ;     ADDR. IS ON U. OR L. DIGIT
 89 0371   CA          JZ   FSTD    ;IF AT UPPER DIGIT
    0372   75
    0373   03
 90 0374   2B          DCX  H       ;DEC. MEMORY ADDR.
 91 0375   E5 FSTD:    PUSH H       ;STORE MEMORY ADDR. FOR
 92                                 ;NEXT TIME.
 93 0376   78          MOV  A,B     ;FETCH CONSTANT ADDR.
 94 0377   C5          PUSH B
 95 0378   CD          CALL CDC     ;CONV. CONST. ADDR. TO BINARY
    0379   C6
    037A   00
 96 037B   C1          POP  B
 97 037C   46          MOV  B,M     ;FETCH EXP. OF CONST.
 98 037D   3E          MVI  A,80    ;TEST SIGN OF
    037E   80
 99 037F   A0          ANA  B       ;     THE DATA
100 0380   CA          JZ   NEG
    0381   88
    0382   03
101 0383   3E          MVI  A,' '   ;IF POSITIVE TYPE ' '
    0384   20
102 0385   C3          JMP  POSI
    0386   8A
    0387   03
103 0388   3E NEG:     MVI  A,2D    ;IF NEGATIVE TYPE '-'
    0389   2D
104 038A   CD POSI:    CALL OPRU
    038B   F8
    038C   03
105 038D   3E          MVI  A,'.'   ;TYPE
    038E   2E
106 038F   CD          CALL OPRU    ;     '.'
    0390   F8
    0391   03
107 0392   2B          DCX  H       ;O/P UPPER TWO
108 0393   CD          CALL MASK    ;    DECIMAL DIGIT
    0394   1A
    0395   04
109 0396   2B          DCX  H       ;O/P LOWER TWO
110 0397   CD          CALL MASK    ;    DECIMAL DIGIT
    0398   1A
    0399   04
111 039A   3E          MVI  A,'E'   ;TYPE
    039B   45
112 039C   CD          CALL OPRU    ;     'E'
    039D   F8
    039E   03
113 039F   3E          MVI  A,40    ;EXTRACT SIGN OF
    03A0   40
114 03A1   A0          ANA  B       ;        EXPONENTIAL
115 03A2   CA          JZ   NEGE
    03A3   AA
    03A4   03
116 03A5   3E          MVI  A,2B    ;TYPE '+'
    03A6   2B
117 03A7   C3          JMP  POSE
    03A8   AC
    03A9   03
118 03AA   3E NEGE:    MVI  A,2D    ;TYPE '-'
    03AB   2D
119 03AC   CD POSE:    CALL OPRU
    03AD   F8
    03AE   03
120 03AF   3E          MVI  A,3F    ;EXTRACT
    03B0   3F
121 03B1   A0          ANA  B       ;        EXPONENTAIL
122 03B2   0C DVDOP:   INR  C       ;CONVERT
123 03B3   D6          SUI  10      ;        EXPONENTIAL
    03B4   10
124 03B5   F2          JP   DVDOP   ;       TO
    03B6   B2
    03B7   03
125 03B8   C6          ADI  10      ;
    03B9   10
126 03BA   0D          DCR  C       ;        B.C.D.
127 03BB   F6          ORI  30      ;CONV. LOWER DIGIT TO ASC111
    03BC   30
128 03BD   47          MOV  B,A     ;STORE LOWER DIGIT IN B
129 03BE   79          MOV  A,C     ;CONV. UPPER DIGIT
130 03BF   F6          ORI  30      ;      TO ASC111
    03C0   30
131 03C1   CD          CALL OPRU    ;O/P UPPER DIGIT OF EXP.
    03C2   F8
    03C3   03
132 03C4   78          MOV  A,B     ;O/P LOWER DIGIT
133 03C5   CD          CALL OPRU    ;          OF EXP.
    03C6   F8
    03C7   03
```

```
134 03C8    3E          MVI  A,' '    ;TYPE
    03C9    20
135 03CA    CD          CALL OPRU     ;     ' '
    03CB    FB
    03CC    03
136 03CD    15          DCR  D        ;DECREMENT DATA COUNTER
137 03CE    C3          JMP  STLG     ;O/P NEXT DATA
    03CF    54
    03D0    03
138             ;SECTION FOR SERVICING DISPLAY AND EXECUTE USER
139             ;PROGRAM OR 3 - TERM CONTROLLER
140 03D1    CD SDPY:    CALL DSET     ;SERVICE DISPLAY
    03D2    72
    03D3    00
141 03D4    2A          LHLD 0BF6     ;INC.
    03D5    F6
    03D6    0B
142 03D7    23          INX  H        ;     SAMPLE NO.
143 03D8    22          SHLD 0BF6     ;     REG.
    03D9    F6
    03DA    0B
144 03DB    3A          LDA  8041     ;I/P STATUS
    03DC    41
    03DD    80
145 03DE    47          MOV  B,A      ;SAVE STATUS
146 03DF    E6          ANI  08       ;TEST AND EXECUTE
    03E0    08
147 03E1    C2          JNZ  TUPR
    03E2    EA
    03E3    03
148 03E4    CD          CALL THTER    ;3-TERM CONTROLLER
    03E5    00
    03E6    10
149 03E7    C3          JMP  TLC
    03E8    F0
    03E9    03
150 03EA    78 TUPR:    MOV  A,B
151 03EB    E6          ANI  40       ;     OR
    03EC    40
152 03ED    CC          CZ   USPR     ;     USER PROGRAM
    03EE    00
    03EF    14
153             ;SECTION FOR TERMINATING DATA LOGGING AND CALCULATION
154             ;SUBROUTINE
155 03F0    F3 TLC:     DI
156 03F1    3E          MVI  A,0F     ;SET FINISH
    03F2    0F
157 03F3    32          STA  0BFE     ;     CALCULATION
    03F4    FE
    03F5    0B
158 03F6    C1          POP  B
159 03F7    C9          RET
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE    9

  1             ;-------------------------------------------------
  2             ;       DATA OUTPUT SUBROUTINE
  3             ;       FUNCTION: TO OUTPUT DATA TO SPECIFIC
  4             ;                 TRANSMITTOR.
  5             ;       INPUTS  : A,C
  6             ;       OUTPUTS : C
  7             ;       DESTROYS: A,C
  8             ;       DESCRIPTION: THIS SUBROUTINE SELECT THE
  9             ;                    O/P DEVICE TX1 OR TX2 BY
 10             ;                    TESTING 1ST BIT OF C. IF
 11             ;                    1ST BIT OF C=0 O/P WOULD BE
 12             ;                    THROUGH TX1 OTHERWISE IT
 13             ;                    WOULD BE TX2.
 14             ;-------------------------------------------------
 15 03FB    B1 OPRU:    ORA  C        ;MIX DATA WITH O/P CONTROL
 16                                   ;BIT
 17 03F9    4F          MOV  C,A
 18 03FA    E6          ANI  80
    03FB    80
 19 03FC    C2          JNZ  OTX2     ;IF O/P TO TX2
    03FD    05
    03FE    04
 20 03FF    CD          CALL SROP1    ;O/P TO TX1 BUFFER
    0400    FB
    0401    01
 21 0402    0E          MVI  C,0      ;CLEAR C
    0403    00
 22 0404    C9          RET
 23 0405    3E OTX2:    MVI  A,7F     ;CLEAR
    0406    7F
 24 0407    A1          ANA  C        ;     1ST BIT
 25 0408    4F          MOV  C,A
 26 0409    CD          CALL SROP2    ;O/P TO TX2 BUFFER
    040A    17
    040B    01
 27 040C    0E          MVI  C,80     ;SET O/P TO TX2
    040D    80
 28 040E    C9          RET
```

```
                        ;-------------------------------------------------
                        ;       TYPE CR & LF SUBROUTINE
                        ;       INPUTS  : C
                        ;       OUTPUTS : C
                        ;       DESTROYS: A
                        ;-------------------------------------------------
040F  3E 0D     CRLF:   MVI  A,0D     ;TYPE
0411  CD F8 03          CALL OPRU     ;     RETURN
0414  3E 0A             MVI  A,0A     ;TYPE
0416  CD F8 03          CALL OPRU     ;     LF
0419  C9                RET
                        ;-------------------------------------------------
                        ;       SUBROUTINE FOR OUTPUTING B.C.D.
                        ;       FUNCTION: TO CONVERT A TWO DIGIT B.C.D. IN
                        ;                 MEMORY TO ASC111 AND O/P.
                        ;       INPUTS  : HL
                        ;       OUTPUTS : NONE
                        ;       DESTROYS: A
                        ;       DESCRIPTION: THE ADDRESS OF THE 2-DIGIT
                        ;                    B.C.D. SHOULD BE PLACED ON
                        ;                    HL.
                        ;-------------------------------------------------
041A  3E F0     MASK:   MVI  A,0F0    ;EXTRACT UPPER
041C  A6                ANA  M        ;        DIGIT
041D  0F                RRC           ;SHIFT
041E  0F                RRC           ;      RIGHT
041F  0F                RRC           ;
0420  0F                RRC           ;      4
0421  C6 30             ADI  30       ;CONV. TO ASC111
0423  27                DAA
0424  FE 40             CPI  40
0426  DA 2A 04          JC   MAS1     ;IF <40
0429  3C                INR  A
042A  CD F8 03  MAS1:   CALL OPRU
042D  3E 0F             MVI  A,0F     ;EXTRACT LOWER
042F  A6                ANA  M        ;        DIGIT
0430  C6 30             ADI  30       ;CONV. TO ASC111
0432  27                DAA
0433  FE 40             CPI  40
0435  DA 39 04          JC   MAS2     ;IF <40
0438  3C                INR  A
0439  CD F8 03  MAS2:   CALL OPRU
043C  C9                RET
```



```
                        ;***********************************************
                        ;
                        ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 7
                        ;        SYSTEM INITIALISATION: TO RESET ALL
                        ;        POINTERS STATUS, AND REPROGRAM ALL
                        ;        COUNTERS, INTERFACINGS.
                        ;
                        ;***********************************************
                        ;RESET BUFFER POINTERS
043D  21 2E 09  INIT:   LXI  H,092E   ;SET O/P BUFFER
0440  22 2C 09          SHLD 092C     ;    POINTER 1
0443  21 92 09          LXI  H,0992   ;SET O/P BUFFER
0446  22 90 09          SHLD 0990     ;    POINTER 2
0449  21 FE 0B          LXI  H,0BFE   ;CLEAR START & SET
044C  36 0F             MVI  M,0F     ;      FINISH REG.
```

```
16                      ;REPROGRAM COUNTERS
17 044E  21             LXI  H,8007  ;LOAD CONTROL WORD
   044F  07
   0450  80
18 0451  36             MVI  M,0B6   ;     TO COUNTER 2
   0452  B6
19 0453  36             MVI  M,76    ;     TO COUNTER 1
   0454  76
20 0455  36             MVI  M,37    ;     TO COUNTER 0
   0456  37
21 0457  21             LXI  H,8006  ;LOAD COUNT
   0458  06
   0459  80
22 045A  36             MVI  M,54    ;     TO
   045B  54
23 045C  36             MVI  M,03    ;     COUNTER 2
   045D  03
24 045E  21             LXI  H,8005  ;LOAD COUNT
   045F  05
   0460  80
25 0461  36             MVI  M,54    ;     TO
   0462  54
26 0463  36             MVI  M,03    ;     COUNTER 1
   0464  03
27 0465  11             LXI  D,8004  ;LOAD
   0466  04
   0467  80
28 0468  2A             LHLD 0800    ;     COUNT
   0469  00
   046A  08
29 046B  EB             XCHG         ;     TO
30 046C  73             MOV  M,E     ;     COUNTER 0
31 046D  72             MOV  M,D     ;     (TIMER INTR.)
32 046E  21             LXI  H,0802  ;SET TIMER ADDR.
   046F  02
   0470  08
33 0471  22             SHLD 0BF2    ;    BUFFER
   0472  F2
   0473  0B
34                      ;REPROGRAM COMMUNICATION INTERFACES
35 0474  21             LXI  H,8011  ;LOAD MODE INSTRUCTION
   0475  11
   0476  80
36 0477  36             MVI  M,0CE   ;     TO TX2
   0478  CE
37 0479  3E             MVI  A,14    ;LOAD COMMAND INSTRUCTION
   047A  14
38 047B  77             MOV  M,A     ;     TO TX2
39 047C  32             STA  0BFC    ;STORE TX2 CONTROL WORD
   047D  FC
   047E  0B
40 047F  21             LXI  H,8009  ;LOAD MODE INSTRUCTION
   0480  09
   0481  80
41 0482  36             MVI  M,0FA   ;     TO TX1
   0483  FA
42 0484  3E             MVI  A,14    ;LOAD COMMAND INSTRUCTION
   0485  14
43 0486  77             MOV  M,A     ;     TO TX1
44 0487  32             STA  0BFD    ;STORE TX1 CONTROL WORD
   0488  FD
   0489  0B
45                      ;RESET INPUT BUFFERS AND REGISTERS
46 048A  21             LXI  H,0     ;CLEAR I/P BUFFERS &
   048B  00
   048C  00
47                                   ;STATUS.
48 048D  22             SHLD 0BF8    ;CLEAR TX2
   048E  F8
   048F  0B
49 0490  22             SHLD 0BFA    ;      TX1
   0491  FA
   0492  0B
50 0493  E5             PUSH H       ;CLEAR
51 0494  D1             POP  D       ;     ALL
52 0495  D5             PUSH D       ;     REGISTER
53 0496  C1             POP  B       ;     AND
54 0497  C5             PUSH B       ;     STATUS
55 0498  F1             POP  PSW     ;
56 0499  FB             EI
```

```
                .TITLE "TELETYPE MONITOR - 8080"
                ;*************************************************
                ;
                ;               TTY MONITOR
                ;               ***********
                ;       ( COMMAND RECONGNIZER )
                ;
                ;       THIS PROGRAM IS A CONTINUATION OF THE
                ;  INTERRUPT SERVICE ROUTINE FOR LEVEL 7.  IT SERVICES
                ;  THE TTY I/P & O/P, PROVIDES PROGRAM I/P, O/P,
                ;  EDITING AND DEBUGGING FACILITIES.
                ;       ALL THE DIFFERENT COMMAND IS EXECUTED BY
                ;  JUMPING TO THE APPROPRIATE ROUTINES.
                ;       WHEN THE ROUTINE FINISHES IT WILL JUMP BACK
                ;  VIA THE REENTRY ROUTE WHICH RESETS THE SP, HENCE
                ;  CLEARING ALL RUBBISH ON THE STACK.
                ;
                ;*************************************************
049A C3         JMP  TYM
049B A7
049C 04
                ;REENTRY TO TTY MONITOR
049D 2A TREN:   LHLD OBF4       ;FETCH STACK ADDR.
049E F4
049F 0B
04A0 F9         SPHL            ;LOAD SP REGISTER
04A1 01         LXI  B,0        ;CLEAR
04A2 00
04A3 00
04A4 11         LXI  D,0        ;       ALL REGISTERS
04A5 00
04A6 00
                ;NORMAL ENTRY TO TTY MONITOR
04A7 3E TYM:    MVI  A,0F       ;PROGRAM INTR. CONTROLLER
04A8 0F
04A9 32         STA  0BFF       ;   TO ACCEPT
04AA FF
04AB 0B
04AC 32         STA  8020       ;   ALL INTR.
04AD 20
04AE 80
04AF 21         LXI  H,0
04B0 00
04B1 00
04B2 39         DAD  SP         ;EXTRACT SP ADDR. TO HL
04B3 22         SHLD 0BF4       ;STORE  SP ADDR.
04B4 F4
04B5 0B
04B6 CD AGAN:   CALL CR         ;TYPE 'CR & LF'
04B7 F9
04B8 0F
04B9 0E         MVI  C,3E       ;TYPE
04BA 3E
04BB CD         CALL SROP2      ;       '>'
04BC 17
04BD 01
04BE CD         CALL SRIP2      ;READ
04BF 5C
04C0 01
04C1 E6         ANI  7F         ;CLEAR PARITY BIT
04C2 7F
04C3 CD         CALL ECHO       ;ECHO TYPE
04C4 6A
04C5 0F
04C6 FE         CPI  'C'
04C7 43
04C8 CA         JZ   COPY       ;IF 'C'
04C9 54
04CA 0D
04CB FE         CPI  'D'
04CC 44
04CD CA         JZ   DUMP       ;IF 'D'
04CE B4
04CF 0D
04D0 FE         CPI  'E'
04D1 45
04D2 CA         JZ   ENTER      ;IF 'E'
04D3 31
04D4 0D
04D5 FE         CPI  'G'
04D6 47
04D7 CA         JZ   GO         ;IF 'G'
04D8 87
04D9 05
04DA FE         CPI  'L'
04DB 4C
04DC CA         JZ   LOAD       ;IF 'L'
04DD 33
04DE 0E
```

```
 48 04ED    FE          CPI  'M'
    04EE    AD
 49 04EF    CA          JZ   MODY     ;IF 'M'
    04F0    F5
    04F1    0C
 50 04F2    FE          CPI  'T'
    04F3    54
 51 04F4    CA          JZ   TRAC     ;IF 'T'
    04F5    C2
    04F6    05
 52 04F7    FE          CPI  'W'
    04F8    57
 53 04F9    CA          JZ   WRI      ;IF 'W'
    04FA    B5
    04FB    0C
 54 04FC    CD          CALL CR       ;TYPE 'CR & LF'
    04FD    F9
    04FE    0F
 55 04FF    0E          MVI  C,'T'    ;TYPE
    0500    54
 56 0501    CD          CALL SROP2    ;
    0502    17
    0503    01
 57 0504    0E          MVI  C,'A'    ;      'TA'
    0505    41
 58 0506    CD          CALL SROP2    ;
    0507    17
    0508    01
 59 0509    C3          JMP  AGAN
    050A    B6
    050B    04
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE   12

  1                ;*************************************************
  2                ;
  3                ;              X - COMMAND
  4                ;          ( IMPLEMENTATION PROGRAM )
  5                ;
  6                ;       THIS PROGRAM ALLOWS THE SPECIFYING OF THE
  7                ;  REGISTERS BEFORE EXECUTING A PROGRAM.
  8                ;       IT IS ATTAINED BY USING A REGISTER BUFFER
  9                ;  IN STORE.
 10                ;       ENTERY:- X P(OR: A,B,C,D,E,H,L,SH,SL) .=TERM.
 11                ;
 12                ;*************************************************
 13 04FE    C5 X:       PUSH B
 14 04FF    CD X1:      CALL TYSP
    0500    DC
    0501    0F
 15 0502    CD          CALL SRIP2    ;READ
    0503    5C
    0504    01
 16 0505    CD          CALL ECHO     ;ECHO TYPE
    0506    6A
    0507    0F
 17 0508    21          LXI  H,09F4   ;LOAD DESTINATE ADDR.
    0509    F4
    050A    09
 18 050B    FE          CPI  'P'
    050C    50
 19 050D    CA          JZ   OPCT     ;IF = 'P'
    050E    65
    050F    05
 20 0510    23          INX  H        ;INC. DESTINATE ADDR.
 21 0511    FE          CPI  'A'
    0512    41
 22 0513    CA          JZ   OPCT
    0514    65
    0515    05
 23 0516    23          INX  H        ;DEST.=09F6
 24 0517    FE          CPI  'C'
    0518    43
 25 0519    CA          JZ   OPCT
    051A    65
    051B    05
 26 051C    23          INX  H        ;DEST.=09F7
 27 051D    FE          CPI  'B'
    051E    42
 28 051F    CA          JZ   OPCT
    0520    65
    0521    05
 29 0522    23          INX  H        ;DEST.=09F8
 30 0523    FE          CPI  'E'
    0524    45
 31 0525    CA          JZ   OPCT
    0526    65
    0527    05
 32 0528    23          INX  H        ;DEST.=09F9
 33 0529    FE          CPI  'D'
    052A    44
 34 052B    CA          JZ   OPCT
    052C    65
    052D    05
```

```
35 052E    23          INX  H       ;INC. DESTINATE ADDR. (09FA)
36 052F    FE          CPI  'L'
   0530    4C
37 0531    CA          JZ   OPCT    ;IF = 'H'
   0532    65
   0533    05
38 0534    23          INX  H       ;INC. DESTINATE ADDR. (09FB)
39 0535    FE          CPI  'H'
   0536    48
40 0537    CA          JZ   OPCT    ;IF = 'L'
   0538    65
   0539    05
41 053A    23          INX  H       ;INC. DESTINATE ADDR. (09FC)
42 053B    FE          CPI  'S'
   053C    53
43 053D    C2          JNZ  TCR     ;IF NOT = 'S'
   053E    51
   053F    05
44 0540    CD          CALL SRIP2   ;READ
   0541    5C
   0542    01
45 0543    CD          CALL ECHO
   0544    6A
   0545    0F
46 0546    FE          CPI  'L'
   0547    4C
47 0548    CA          JZ   OPCT    ;IF = 'H'
   0549    65
   054A    05
48 054B    23          INX  H       ;INC. DESTINATE ADDR. (09FD)
49 054C    FE          CPI  'H'
   054D    48
50 054E    CA          JZ   OPCT    ;IF = 'L'
   054F    65
   0550    05
51 0551    FE TCR:     CPI  '.'
   0552    2E
52 0553    C1          POP  B
53 0554    C8          RZ           ;IF '.' RETURN
54 0555    CD          CALL TYSP    ;TYPE ' '
   0556    DC
   0557    0F
55 0558    3E          MVI  A,'R'   ;TYPE
   0559    52
56 055A    CD          CALL ECHO    ;
   055B    6A
   055C    0F
57 055D    3E          MVI  A,'B'   ;   'RB'
   055E    42
58 055F    CD          CALL ECHO    ;
   0560    6A
   0561    0F
59 0562    C3          JMP  X
   0563    FE
   0564    04
60 0565    7E OPCT:    MOV  A,M     ;O/P
61 0566    EB          XCHG         ;    REG.
62 0567    6F          MOV  L,A     ;    CONTENT
63 0568    CD          CALL OPSL    ;
   0569    8B
   056A    0F
64 056B    EB          XCHG         ;
65 056C    CD          CALL TYSP    ;TYPE ' '
   056D    DC
   056E    0F
66 056F    CD IPCT:    CALL RASC    ;I/P 1ST DIGIT
   0570    50
   0571    0F
67 0572    D2          JNC  IPSED   ;IF = HEX
   0573    7D
   0574    05
68 0575    FE          CPI  0D
   0576    0D
69 0577    CA          JZ   X1      ;IF = 'CR'
   0578    FF
   0579    04
70 057A    C3          JMP  TCR
   057B    51
   057C    05
71 057D    CD IPSED:   CALL INAD3   ;LOAD & I/P 2ND DIGIT IN E
   057E    27
   057F    0F
72 0580    DA          JC   TCR
   0581    51
   0582    05
73 0583    73          MOV  M,E     ;LOAD I/P TO MEMORY
74 0584    C3          JMP  X1
   0585    FF
   0586    04
```

```
                ;**************************************************
                ;
                ;            G - GO COMMAND
                ;       (IMPLEMENTATION PROGRAM)
                ;
                ;     THIS PROGRAM WOULD CLEAR ALL REGISTERS AND
                ;  JUMP TO THE ADDRESS SPECIFIED.  THE REGISTERS
                ;  COULD BE SPECIFIED BY THE X - COMMAND.
                ;
                ;**************************************************
0587  CD  GO:      CALL TYSP     ;TYPE ' '
0588  DC
0589  0F
058A  CD          CALL CLX      ;CLEAR 09F4 TO 09FD
058B  CF
058C  0F
058D  CD          CALL INAD     ;I/P START ADDR. ON DE
058E  03
058F  0F
0590  42          MOV  B,D      ;STORE ADDR.
0591  4B          MOV  C,E      ;      IN BC
0592  DA          JC   TICR
0593  9E
0594  05
0595  CD          CALL TYSP
0596  DC
0597  0F
0598  CD          CALL SRIP2    ;READ
0599  5C
059A  01
059B  CD  RERE:   CALL ECHO     ;ECHO
059C  6A
059D  0F
059E  FE  TICR:   CPI  0D
059F  0D
05A0  CA          JZ   XGO      ;IF 'CR'
05A1  AB
05A2  05
05A3  FE          CPI  'X'
05A4  58
05A5  C2          JNZ  RERE     ;IF NOT = 'X'
05A6  9B
05A7  05
05A8  CD          CALL X
05A9  FE
05AA  04
05AB  2A  XGO:    LHLD 09FC     ;FETCH STACK CONTENT
05AC  FC
05AD  09
05AE  E5          PUSH H        ;POSITION STACK
05AF  C5          PUSH B        ;LOAD START ADDR.
05B0  2A          LHLD 09F4     ;FETCH PSW
05B1  F4
05B2  09
05B3  E5          PUSH H
05B4  2A          LHLD 09F6     ;FETCH BC
05B5  F6
05B6  09
05B7  44          MOV  B,H      ;POSITION
05B8  4D          MOV  C,L      ;         BC
05B9  2A          LHLD 09F8     ;FETCH DE
05BA  F8
05BB  09
05BC  EB          XCHG          ;POSITION DE
05BD  2A          LHLD 09FA     ;FETCH HL
05BE  FA
05BF  09
05C0  F1          POP  PSW      ;POSITION PSW
05C1  C9          RET           ;JUMP TO START ADDR.
```

```
                ;**********************************************
                ;       OUTPUT SUBROUTINE FOR OUTPUTTING TO TX1
                ;       FUNCTION: TO PLACE A OUTPUT CHARACTER ON
                ;                 THE OUTPUT BUFFER OF TX1
                ;       INPUTS  : C
                ;       OUTPUTS : NONE
                ;       DISTROYS: NOTHING
                ;       DESCRIPTION; TO ENTER THIS ROUTINE, 1ST PUT
                ;                    O/P CHARACTER ON REG. C. THEN
                ;                    CALL SROP1
                ;**********************************************
01FB    F5  SROP1:  PUSH PSW
01FC    E5          PUSH H
01FD    2A  WAIT1:  LHLD 092C     ;HL_BUFFER PTER.
01FE    2C
01FF    09
0200    3E          MVI  A,8E     ;TEST
0201    8E
0202    95          SUB  L        ;IF BUFFER PTER.>LIMIT
0203    FA          JM   WAIT1
0204    FD
0205    01
0206    F3          DI
0207    71          MOV  M,C      ;BUFFER_C
0208    23          INX  H        ;INCREMENT HL
0209    22          SHLD 092C     ;STORE BUFFER PTER.
020A    2C
020B    09
                ;SECTION FOR ENABLING TRANSMISSION
020C    3A          LDA  0BFD     ;FETCH TX1 CONTROL WORD
020D    FD
020E    0B
020F    E6          ANI  01
0210    01
0211    C2          JNZ  ENAB1    ;IF TX1 ENABLED
0212    1F
0213    02
0214    3A          LDA  0BFD     ;FETCH TX1 CONTROL WORD
0215    FD
0216    0B
0217    F6          ORI  21       ;SET TX REDY. FLAG
0218    21
0219    32          STA  0BFD     ;STORE TX1 CONTROL WORD
021A    FD
021B    0B
021C    32          STA  8009     ;PROGRAM TX1
021D    09
021E    80
021F    FB  ENAB1:  EI
0220    E1          POP  H        ;END OF SUBROUTINE
0221    F1          POP  PSW
0222    C9          RET
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE    6

                ;**********************************************
                ;
                ;       SERVICE ROUTINE FOR INTERRUPT LEVEL 5
                ;        TO INPUT A CHARACTER VIA SERIAL
                ;        TRANSMISSION INTERFACE 1 (RX1)
                ;
                ;**********************************************
                ;SECTION FOR SAVING ALL STATUS AFTER INTR.
0223    D5  IN3:    PUSH D
0224    C5          PUSH B
0225    7E          MOV  A,M
0226    F5          PUSH PSW
0227    3E          MVI  A,02
0228    02
0229    77          MOV  M,A
022A    32          STA  8020
022B    20
022C    80
                ;THIS SECTION READS THE DATA AND ORGANIZE ALL
                ;FLAGS
022D    21          LXI  H,8009   ;LOAD ADDR. OF TX1 STATUS REG.
022E    09
022F    80
0230    11          LXI  D,0BFB   ;LOAD ADDR. OF RX1 STATUS BUF.
0231    FB
0232    0B
0233    7E          MOV  A,M      ;READ TX1 STATUS WORD.
0234    EB          XCHG
0235    77          MOV  M,A      ;STORE STATUS IN BUFFER
                ;DISABLE READ & DTR
0236    3A          LDA  0BFD     ;READ TX1 CONTROL WORD
0237    FD
0238    0B
0239    E6          ANI  0F9      ;CLEAR RX RDY & DTR
023A    F9
023B    32          STA  0BFD     ;STORE TX1 CONTROL WORD
023C    FD
023D    0B
023E    32          STA  8009     ;PROGRAM TX2
023F    09
0240    80
```

```
                        ;INPUT DATA
0241  EB                XCHG
0242  2B                DCX  H        ;SET I/P PORT ADDR.
0243  1B                DCX  D        ;SET I/P BUFFER ADDR.
0244  7E                MOV  A,M      ;I/P DATA
0245  FB                EI
0246  EB                XCHG
0247  77                MOV  M,A      ;STORE DATA IN BUFFER
0248  C3                JMP  INTRE    ;END INTR. SERVICE ROUTINE
0249  55
024A  00
                        ;*************************************************
                        ;       INPUT SUBROUTINE FOR INPUTTING FROM TX1
                        ;       FUNCTION: TO READ A CHARACTER FROM THE
                        ;                 I/P BUFFER
                        ;       INPUTS  : NONE
                        ;       OUTPUTS : ACC, RESETS I/P BUFFER & STATUS
                        ;       CALLS   : SROP2
                        ;       DISTROYS: NOTHING
                        ;       DESCRIPTIONS: THIS ROUTINE PUTS A 7 BIT
                        ;                     DATA ON THE ACCUMULATOR WITH
                        ;                     1ST BIT (8TH BIT) EQUAL TO '0'.
                        ;                     THIS ROUTINE WOULD ALSO CHECK
                        ;                     FLAMING ERROR AND PARITY ERROR
                        ;                     'FERROR' = FLAMING ERROR
                        ;                     'PERROR' = PARITY ERROR
                        ;                     IF PARITY ERROR - 1ST BIT
                        ;                     (M.S.B.) OF DATA WOULD BE SET
                        ;                     TO '1'.
                        ;                     NOTE * ALL ERROR ARE OUTPUTTED
                        ;                     VIA TX2
                        ;*************************************************
024B  E5  SRIP1:  PUSH H
024C  C5          PUSH B
024D  2A  RDY1:   LHLD OBFA     ;FETCH DATA & STATUS
024E  FA
024F  0B
0250  7C          MOV  A,H      ;ACC._STATUS
0251  E6          ANI  2        ;TEST IF RX RDY IS SET
0252  02
0253  CA          JZ   RDY1     ;IF READY
0254  4D
0255  02
0256  3E  CLR1:   MVI  A,0
0257  00
0258  32          STA  OBFB     ;CLEAR STATUS BUFFER
0259  FB
025A  0B
                        ;SECTION FOR ENABLING READ, DTR & ERROR RESET
025B  3A          LDA  OBFD     ;FETCH CONTROL WORD
025C  FD
025D  0B
025E  F6          ORI  16
025F  16
0260  32          STA  OBFD     ;STORE CONTROL WORD
0261  FD
0262  0B
0263  32          STA  8009     ;READ & DTR ENABLED
0264  09
0265  80
                        ;ERROR CHECK
0266  7D          MOV  A,L      ;CLEAR
0267  E6          ANI  7F       ;    PARITY
0268  7F
0269  6F          MOV  L,A      ;    BIT
026A  7C          MOV  A,H      ;ACC._STATUS
026B  E6          ANI  8        ;TEST IF PARITY ERROR
026C  08
026D  CA          JZ   FER1     ;IF NOT
026E  77
026F  02
0270  7D          MOV  A,L      ;SET 1ST BIT
0271  F6          ORI  80       ;    OF DATA '1'
0272  80
0273  6F          MOV  L,A      ;    IF PARITY ERROR
0274  CD          CALL PER2     ;TYPE 'PERROR' VIA TX2
0275  A6
0276  01
0277  7C  FER1:   MOV  A,H
0278  E6          ANI  20       ;IF FRAMING ERROR
0279  20
027A  CA          JZ   EIP1     ;IF NOT
027B  85
027C  02
027D  0E          MVI  C,'F'    ;TYPE
027E  46
027F  CD          CALL SROP2    ;    'FERROR'
0280  17
0281  01
0282  CD          CALL ER2
0283  AB
0284  01
```

```
                ;*****************************************************
                ;
                ;           T - TRACE ERROR
                ;       ( IMPLEMENTATION PROGRAM )
                ;
                ;       NAME: TRAC
                ;       THIS ROUTINE HAVE TWO WAYS OF ENTRY:-
                ;       (1) >T START ADDR. TERM. ADDR.(AND/OR N X ...)
                ;       THIS EXECUTE THE INSTRUCTION STARTING FROM
                ;  THE START ADDRESS. IT ALSO PRINT OUT PC, INSTRUCTION
                ;  AND ALL REGISTERS & STATUS ON COMPLETION OF EACH
                ;  INSTRUCTION IN THE FORM:-
                ;  0000 II IIII AASW BBCC DDEE HHLL SHSL SPAD
                ;  WHERE: 0000= PROGRAM COUNTER
                ;         IIII= INSTRUCTION
                ;         AA  = ACCUMULATOR
                ;         SPAD= CONTENT OF STACK POINTER
                ;       IF N IS TYPED:- NO O/P EXCEPT LAST STEP
                ;       (2) >T START ADDR. S (AND/OR X ...)
                ;       THIS WILL EXECUTE ONE INSTRUCTION FROM THE
                ;  START ADDRESS AND PRINT OUT THE PC, INSTRUCTION,
                ;  REGISTERS ETC. AS IN (1). IT THEN WAIT FOR A CR
                ;  BEFORE EXECUTING THE NEXT INSTRUCTION.
                ;       NOTE: THE X COMMAND CAN BE USED TO CHANGE THE
                ;       REGISTERS AT EACH INSTRUCTION CYCLE.
                ;
                ;       CALLS: INAD, RASC, OPSA, X, TYSP, ECHO, XGO
                ;
                ;*****************************************************
                ;ALLOW THE INPUTTING OF ADDRESSES
05C2 CD 05C3 DC TRAC:   CALL TYSP       ;TYPE ' '
05C4 0F
05C5 CD         CALL CLX        ;CLEAR 09F4 TO 09FD
05C6 CF
05C7 0F
05C8 67         MOV  H,A        ;CLEAR 'N' REG. AND
05C9 6F         MOV  L,A
05CA 22         SHLD 0A0C       ;   BYTE COUNTER
05CB 0C
05CC 0A
05CD CD         CALL INAD       ;I/P START ADDR.
05CE 03
05CF 0F
05D0 EB         XCHG
05D1 22         SHLD 0A00       ;SAVE START ADDR.
05D2 00
05D3 0A
05D4 3E         MVI  A,' '
05D5 20
05D6 CD 05D7 6A TC:     CALL ECHO
05D8 0F
05D9 CD 05DA 50 T1:     CALL RASC       ;I/P ONE CHARACTER
05DB 0F
05DC D2         JNC  CAID       ;IF HEX.
05DD ED
05DE 05
05DF FE         CPI  'S'
05E0 53
05E1 C2         JNZ  T1         ;IF NOT= 'S'
05E2 D9
05E3 05
05E4 21         LXI  H,0        ;CLEAR 0A02
05E5 00
05E6 00
05E7 22         SHLD 0A02       ;       AND 0A03
05E8 02
05E9 0A
05EA C3         JMP  ENCM
05EB F7
05EC 05
05ED CD 05EE 09 CAID:   CALL INAD1      ;I/P TERMINATE ADDR.
05EF 0F
05F0 EB         XCHG
05F1 22         SHLD 0A02       ;STORE TERM. ADDR. IN BUFFER
05F2 02
05F3 0A
05F4 DA         JC   TC1        ;IF LAST CHARACTER NOT HEX.
05F5 00
05F6 06
05F7 CD 05F8 DC ENCM:   CALL TYSP       ;TYPE ' '
05F9 0F
05FA CD 05FB 5C T2:     CALL SRIP2      ;READ CR OR X
05FC 01
05FD CD         CALL ECHO       ;ECHO TYPE
05FE 6A
05FF 0F
```

```
 55 0600    FE TC1:    CPI  0D
    0601    0D
 56 0602    CA         JZ   ROUN    ;IF 'CR' START EXECUTING
    0603    15
    0604    06
 57 0605    FE         CPI  'N'
    0606    4E
 58 0607    C2         JNZ  TC2
    0608    0D
    0609    06
 59 060A    32         STA  0A0C    ;SET 'N' REG.
    060B    0C
    060C    0A
 60 060D    FE TC2:    CPI  'X'
    060E    58
 61 060F    C2         JNZ  ENCH    ;IF NOT 'X' READ AGAIN
    0610    F7
    0611    05
 62 0612    CD         CALL X       ;X COMMAND
    0613    FE
    0614    04
 63 0615    3E ROUN:   MVI  A,0C3
    0616    C3
 64 0617    32         STA  0A07    ;LOAD JMP
    0618    07
    0619    0A
 65 061A    21         LXI  H,BACK  ;LOAD
    061B    C9
    061C    07
 66 061D    22         SHLD 0A08    ;      BACK
    061E    08
    061F    0A
 67 0620    2A         LHLD 0A00    ;FETCH PC
    0621    00
    0622    0A
 68 0623    22         SHLD 0A0E    ;LOAD DISP. BUFFER
    0624    0E
    0625    0A
 69 0626    7E         MOV  A,M     ;MOVE 1ST BYE OF INST.
 70 0627    32         STA  0A04    ;      TO BUFFER
    0628    04
    0629    0A
 71 062A    EB         XCHG         ;STORE PC IN DE
 72 062B    21         LXI  H,0     ;CLEAR 0A05 & 0A06 I.E.
    062C    00
    062D    00
 73 062E    22         SHLD 0A05    ;      2ND & 3RD BYTE
    062F    05
    0630    0A
 74 0631    EB         XCHG         ;RESTORE PC TO HL
 75 0632    23         INX  H       ;INC. PC
TELETYPE MONITOR - 8080          CALOS-80 V2.12 00/00/70 PAGE   14

  1                ;***********************************************
  2                ;       THE FOLLOWING SECTION SORT OUT THE 1 BYTE,
  3                ;  2 BYTE AND 3 BYTE INSTRUCTIONS. IT ALSO SORT OUT
  4                ;  INSTRUCTIONS WHICH REQUIRE THE CHANGING OF THE
  5                ;  PC.
  6                ;***********************************************
  7 0633    57         MOV  D,A     ;SAVE INSTR.
  8 0634    E6         ANI  80
    0635    80
  9 0636    C2         JNZ  SEO     ;IF D7=1
    0637    69
    0638    06
 10 0639    7A         MOV  A,D
 11 063A    E6         ANI  40
    063B    40
 12 063C    CA         JZ   T4      ;IF INST. = 00XXX XXX
    063D    42
    063E    06
 13 063F    C3         JMP  ONEB    ;IF INST. = 01XXX XXX
    0640    C3
    0641    06
 14 0642    7A T4:     MOV  A,D
 15 0643    E6         ANI  07
    0644    07
 16 0645    FE         CPI  06
    0646    06
 17 0647    C2         JNZ  T5      ;IF INST. NOT= 00XXX110
    0648    4D
    0649    06
 18 064A    C3         JMP  TWOB
    064B    CC
    064C    06
 19 064D    FE T5:     CPI  01
    064E    01
 20 064F    C2         JNZ  T6      ;IF INST. NOT = 00XXX001
    0650    5B
    0651    06
 21 0652    7A         MOV  A,D
 22 0653    E6         ANI  08
    0654    08
```

```
23 0655    CA          JZ   THRD    ;IF INST. = 00XX0001
   0656    DC
   0657    06
24 0658    C3          JMP  ONEB    ;IF INST. =00XX1001
   0659    C3
   065A    06
25 065B    FE T4:      CPI  02
   065C    02
26 065D    C2          JNZ  ONEB    ;IF INST. NOT = 00XXX010
   065E    C3
   065F    06
27 0660    7A          MOV  A,D
28 0661    E6          ANI  20
   0662    20
29 0663    CA          JZ   ONEB    ;IF INST. = 000XX010
   0664    C3
   0665    06
30 0666    C3          JMP  THRB    ;IF INST. = 001XX010
   0667    DC
   0668    06
31 0669    7A SED:     MOV  A,D
32 066A    E6          ANI  40
   066B    40
33 066C    CA          JZ   ONEB    ;IF INST. = 10XXXXXX
   066D    C3
   066E    06
34 066F    7A          MOV  A,D
35 0670    E6          ANI  07
   0671    07
36 0672    FE          CPI  02
   0673    02
37 0674    CA          JZ   JCON    ;IF INST. =11XXX010
   0675    02
   0676    07
38 0677    FE          CPI  04
   0678    04
39 0679    CA          JZ   CCON    ;IF INST. = 11XXX100
   067A    46
   067B    07
40 067C    FE          CPI  06
   067D    06
41 067E    CA          JZ   TWOB    ;IF INST. = 11XXX110
   067F    CC
   0680    06
42 0681    FE          CPI  00
   0682    00
43 0683    CA          JZ   RCON    ;IF INST. =11XXX000
   0684    7C
   0685    07
44 0686    FE          CPI  07
   0687    07
45 0688    CA          JZ   RSTA    ;IF INST. = 11XXX111
   0689    AF
   068A    07
46 068B    FE          CPI  05
   068C    05
47 068D    C2          JNZ  T7      ;IF NOT = 11XXX101
   068E    99
   068F    06
48 0690    7A          MOV  A,D
49 0691    E6          ANI  08
   0692    08
50 0693    CA          JZ   ONEB    ;IF = 11XX0101
   0694    C3
   0695    06
51 0696    C3          JMP  CALLI   ;IF = 11001101
   0697    27
   0698    07
52 0699    FE T7:      CPI  03
   069A    03
53 069B    C2          JNZ  T8      ;IF INST NOT = 11XXX011
   069C    AE
   069D    06
54 069E    7A          MOV  A,D
55 069F    E6          ANI  38
   06A0    38
56 06A1    CA          JZ   JMPA    ;IF INST. = 11000 011
   06A2    F0
   06A3    06
57 06A4    E6          ANI  30
   06A5    30
58 06A6    FE          CPI  10
   06A7    10
59 06A8    CA          JZ   TWOB    ;IF INST. = 1101X011
   06A9    CC
   06AA    06
60 06AB    C3          JMP  ONEB    ;IF NOT 1 BYTE
   06AC    C3
   06AD    06
61 06AE    FE T8:      CPI  01
   06AF    01
62 06B0    C2          JNZ  ONEB    ;IF NOT = 11XXX001
   06B1    C3
   06B2    06
```

```
 63 06BB    7A          MOV  A,D
 64 06BC    E6          ANI  38
    06BD    38
 65 06BE    FE          CPI  08
    06BF    08
 66 06C0    CA          JZ   RETA    ;IF INST. = 11001001
    06C1    70
    06C2    07
 67 06C3    FE          CPI  28
    06C4    28
 68 06C5    CA          JZ   PCHLI   ;IF INST.= 11101001
    06C6    A6
    06C7    07
 69 06C8    C3          JMP  ONEB
    06C9    C3
    06CA    06
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE   15

  1                 ;***********************************************
  2                 ;       THE FOLLOWING SECTION TAKES CARE OF THE
  3                 ;  ONE BYTE INSTRUCTION OTHER THAN THOSE WHICH
  4                 ;  INCORPORATED A 'JUMP'.
  5                 ;       THE PART STARTING WITH SPC IS THE PART
  6                 ;  FOR EXECUTING THE ONE INSTRUCTION IN THE BUFFER.
  7                 ;***********************************************
  8 06CB    22 ONEB:  SHLD 0A00    ;STORE PC IN BUFFER
    06CC    00
    06CD    0A
  9 06CE    01        LXI  B,0A04  ;LOAD INST. ADDR. IN BC
    06CF    04
    06D0    0A
 10 06D1    C3        JMP  XGO     ;JMP TO EXECUTE
    06D2    AB
    06D3    05
 11                 ;***********************************************
 12                 ;       THE FOLLOWING SECTION HANDLES THE
 13                 ;  2 BYTE INSTRUCTIONS.
 14                 ;***********************************************
 15 06D4    7E TWOB:  MOV  A,M     ;FETCH 2ND BYTE OF INST.
 16 06D5    32        STA  0A05    ;STORE IN BUFFER
    06D6    05
    06D7    0A
 17 06D8    32        STA  0A10    ;    AND DISP.
    06D9    10
    06DA    0A
 18 06DB    3E        MVI  A,01    ;SET
    06DC    01
 19 06DD    32        STA  0A0D    ;   1 BYTE
    06DE    0D
    06DF    0A
 20 06E0    23        INX  H       ;INC. PC
 21 06E1    C3        JMP  ONEB
    06E2    C3
    06E3    06
 22                 ;***********************************************
 23                 ;       THE FOLLOWING SECTION HANDLES THE
 24                 ;  3 BYTE INSTRUCTIONS.
 25                 ;***********************************************
 26 06E4    5E THRB:  MOV  E,M     ;FETCH 2ND BYTE OF INST.
 27 06E5    23        INX  H       ;INC. P.C.
 28 06E6    56        MOV  D,M     ;FETCH 3RD BYTE OF INST.
 29 06E7    23        INX  H       ;INC. P.C.
 30 06E8    EB        XCHG         ;STORE P.C.
 31 06E9    22        SHLD 0A05    ;STORE 2ND & 3RD BYTE IN
    06EA    05
    06EB    0A
 32 06EC    22        SHLD 0A10    ;  BUFFER & DISP.
    06ED    10
    06EE    0A
 33 06EF    3E        MVI  A,02    ;SET
    06F0    02
 34 06F1    32        STA  0A0D    ;   3 BYTE
    06F2    0D
    06F3    0A
 35 06F4    EB        XCHG
 36 06F5    C3        JMP  ONEB
    06F6    C3
    06F7    06
 37                 ;***********************************************
 38                 ;       THE FOLLOWING SECTION TAKES CARE OF THE
 39                 ;  JMP INSTRUCTION.
 40                 ;***********************************************
 41 06F8    5E JMPA:  MOV  E,M     ;FETCH 2ND BYTE OF INST.
 42 06F9    23        INX  H       ;INC P.C.
 43 06FA    56        MOV  D,M     ;FETCH 3RD BYTE OF INST.
 44 06FB    EB        XCHG         ;HL_DE
 45 06FC    22        SHLD 0A00    ;TRANSFER ADDR. TO PC BUFFER
    06FD    00
    06FE    0A
 46 06FF    22        SHLD 0A10    ;   AND DISP.
    0700    10
    0701    0A
```

```
47 06FA   3E          MVI  A,02     ;SET
   06FB   02
48 06FC   32          STA  0A0D     ;   3 BYTE
   06FD   0D
   06FE   0A
49 06FF   C3          JMP  OPRG
   0700   F1
   0701   07
50                    ;*************************************************
51                    ;       THE FOLLOWING SECTION TAKES CARE OF THE
52                    ;  CONDITIONAL JUMP INSTRUCTIONS.
53                    ;*************************************************
54 0702   5E JCON:    MOV  E,M      ;FETCH 2ND BYTE OF INST.
55 0703   23          INX  H        ;INC. P.C.
56 0704   56          MOV  D,M      ;FETCH 3RD BYTE OF INST.
57 0705   23          INX  H        ;INC. P.C.
58 0706   EB          XCHG          ;STORE P.C.
59 0707   22          SHLD 0A0A     ;LOAD ADDR. IN PC BUFFER 2
   0708   0A
   0709   0A
60 070A   22          SHLD 0A10     ;   AND DISP.
   070B   10
   070C   0A
61 070D   3E          MVI  A,02     ;SET
   070E   02
62 070F   32          STA  0A0D     ;   3 BYTE
   0710   0D
   0711   0A
63 0712   21          LXI  H,YJC    ;LOAD ADDR. OF YJC TO
   0713   1C
   0714   07
64 0715   22          SHLD 0A05     ;     CONDN. JUMP.
   0716   05
   0717   0A
65 0718   EB          XCHG          ;RESTORE P.C.
66 0719   C3          JMP  ONEB
   071A   C3
   071B   06
67 071C   E5 YJC:     PUSH H
68 071D   2A          LHLD 0A0A     ;FETCH JUMP TO ADDR.
   071E   0A
   071F   0A
69 0720   22          SHLD 0A00     ;LOAD ADDR. TO PC BUFFER
   0721   00
   0722   0A
70 0723   E1          POP  H
71 0724   C3          JMP  BACK
   0725   C9
   0726   07
TELETYPE MONITOR - 8080          CALOS-80 V2.12 00/00/70 PAGE   16

 1                    ;*************************************************
 2                    ;       THE FOLLOWING SECTION HANDLES THE CALL
 3                    ;  INSTRUCTION.
 4                    ;*************************************************
 5 0727   EB CALLI:   XCHG          ;SAVE PC
 6 0728   2A          LHLD 09FC     ;FETCH SP FROM BUFFER
   0729   FC
   072A   09
 7 072B   E5          PUSH H        ;PUSH PREVIOUS SP ONTO STACK
 8 072C   EB          XCHG          ;RESTORE PC
 9 072D   5E          MOV  E,M      ;FETCH 2ND BYTE OF INST.
10 072E   23          INX  H        ;INC. PC
11 072F   56          MOV  D,M      ;FETCH 3RD BYTE OF INST.
12 0730   23          INX  H        ;INC. PC
13 0731   22          SHLD 09FC     ;LOAD P.C. ONTO SP
   0732   FC
   0733   09
14 0734   EB          XCHG
15 0735   22          SHLD 0A00     ;LOAD JMP TO ADDR. IN
   0736   00
   0737   0A
16 0738   22          SHLD 0A10     ;   PC BUFFER AND DISP.
   0739   10
   073A   0A
17 073B   3E          MVI  A,02     ;SET
   073C   02
18 073D   32          STA  0A0D     ;   3 BYTES
   073E   0D
   073F   0A
19 0740   21          LXI  H,09FE   ;SET UP ADDR. OF
   0741   FE
   0742   09
20                                  ;    SP REGISTER BUFFER
21 0743   C3          JMP  LSPR
   0744   E7
   0745   07
22                    ;*************************************************
23                    ;       THE FOLLOWING SECTION HANDLES THE
24                    ;  "CONDITIONAL CALL" INSTRUCTIONS.
25                    ;*************************************************
26 0746   5E CCON:    MOV  E,M      ;FETCH 2ND BYTE
27 0747   23          INX  H
```

```
                        56          MOV  D,M     ;FETCH 3RD BYTE
                        23          INX  H
                        EB          XCHG
                        22          SHLD 0A0A    ;LOAD ADDR. IN PC BUFFER 2
                        0A
                        0A
                        22          SHLD 0A10    ;   AND DISP.
                        10
                        0A
                        3E          MVI  A,02    ;SET
                        02
                        32          STA  0A0D    ;   3 BYTES
                        0D
                        0A
                        21          LXI  H,YCC   ;LOAD YCC TO ADDR. OF
                        60
                        07
                        22          SHLD 0A05    ;     CCOND BUFFER
                        05
                        0A
                        EB          XCHG         ;RESTORE ORIGINAL PC
                        C3          JMP  ONEB
                        C3
                        06
                    ;IF THE 'CONDITION' IS SATISFIED IT SHALL JUMP TO
                    ;THE FOLLOWING ROUTINE
                    E3 YCC:     XTHL
                        2A          LHLD 0A00    ;FETCH ORIGINAL PC
                        00
                        0A
                        E3          XTHL         ;PUT ORIGINAL PC
                        E5          PUSH H       ;    ONTO PROPER
                                                 ;    LOCATION OF SP
                        2A          LHLD 0A0A    ;FETCH JUMP TO ADDR.
                        0A
                        0A
                        22          SHLD 0A00    ;LOAD JUMP TO ADDR.
                        00
                        0A
                                                 ;     ONTO PC
                        E1          POP  H       ;RESTORE HL
                        C3          JMP  BACK
                        C9
                        07
                    ;*************************************************
                    ;       THE FOLLOWING SECTION HANDLES THE RET
                    ;  INSTRUCTION.
                    ;*************************************************
                    2A RETA:    LHLD 09FC    ;FETCH RETURN ADDR.
                        FC
                        09
                                                 ;      FROM SP BUFFER
                        22          SHLD 0A00    ;LOAD ADDR. ONTO PC BUFFER
                        00
                        0A
                        21          LXI  H,09FC  ;SET SP BUFFER ADDR.
                        FC
                        09
                        C3          JMP  LSP
                        E2
                        07
                    ;*************************************************
                    ;       THE FOLLOWING SECTION HANDLES THE
                    ;  "CONDITIONAL RETURN" INSTRUCTION.
                    ;*************************************************
                    E5 RCON:    PUSH H       ;SAVE PC
                        2A          LHLD 09FC    ;FETCH RETURN ADDR.
                        FC
                        09
                                                 ;      FROM SP BUFFER
                        22          SHLD 0A0A    ;STORE IN PC BUFFER 2
                        0A
                        0A
                        21          LXI  H,YRC   ;LOAD YRC
                        93
                        07
                        22          SHLD 09FC    ;     ONTO SP BUFFER
                        FC
                        09
                        21          LXI  H,NRC   ;LOAD NRC ONTO JMP TO
                        9E
                        07
                        22          SHLD 0A08    ;     ADDR. OF EXIT
                        08
                        0A
                        E1          POP  H       ;RESTORE PC
                        C3          JMP  ONEB
                        C3
                        06
```

```
                        ;IF THE 'CONDITION' IS SATISFIED IT WILL JMP TO
                        ;THIS ROUTINE.
07??    E5      YRC:    PUSH H          ;SAVE HL
07??    2A              LHLD 0A0A       ;FETCH RET ADDR.
07??    0A
07??    0A
07??    22              SHLD 0A00       ;LOAD RET ADDR. ONTO PC.
07??    00
07??    0A
                                        ;       BUFFER
07??    E1              POP  H          ;RESTORE HL
07??    C3              JMP  BACK
07??    C9
07??    07
                        ;IF THE CONDITION IS NOT SATISFIED IT WILL JMP TO
                        ;THIS ROUTINE.
07??    E3      NRC:    XTHL            ;EXTRACT YRC FROM SP
07??    2A              LHLD 0A0A       ;FETCH RET ADDR.
07??    0A
07??    0A
07??    E3              XTHL            ;PUT BACK ADDR. ONTO SP
07??    C3              JMP  BACK
07??    C9
07??    07
TELETYPE MONITOR - 8080                 CALOS-80 V2.12 00/00/70 PAGE   17

                        ;***********************************************
                        ;       THE FOLLOWING SECTION HANDLES THE PCHL
                        ;  (JUMP H AND L INDIRECT - MOVE HL TO PC)
                        ;  INSTRUCTION.
                        ;***********************************************
07??    2A      PCHLI:  LHLD 09FA       ;FETCH HL FROM BUFFER
07??    FA
07??    09
07??    22              SHLD 0A00       ;LOAD HL ONTO PC BUFFER
07??    00
07??    0A
07??    C3              JMP  OPRG
07??    F1
07??    07
                        ;***********************************************
                        ;       THE FOLLOWING SECTION HANDLES THE 'RST'
                        ;  (RESTART) INSTRUCTION.
                        ;***********************************************
07??    EB      RSTA:   XCHG            ;SAVE PC
07??    2A              LHLD 09FC       ;FETCH SP FROM BUFFER
07??    FC
07??    09
07??    E5              PUSH H          ;PUSH SP BUFFER ONTO
                                        ;       SP
07??    EB              XCHG            ;RESTORE PC
07??    22              SHLD 09FC       ;PUT PC ONTO SP BUFFER
07??    FC
07??    09
07??    3A              LDA  0A04       ;FETCH INST. FROM BUFFER
07??    04
07??    0A
07??    E6              ANI  38         ;EXTRACT JMP TO ADDR.
07??    38
07??    6F              MOV  L,A        ;ASSEMBLE JMP TO
07??    26              MVI  H,0        ;           ADDR.
07??    00
07??    22              SHLD 0A00       ;LOAD ADDR. ONTO PC BUFFER
07??    00
07??    0A
07??    21              LXI  H,09F      ;SET UP SP REGISTER ADDR.
07??    9F
07??    00
07??    C3              JMP  LSPR
07??    E7
07??    07
                        ;***********************************************
                        ;       THE FOLLOWING SECTION CONSTITUTE THE
                        ;  LAST PART OF THE ROUTINE WHICH (1) PUT ALL
                        ;  REGISTERS & STATUS ETC INTO BUFFER.  (2) O/P
                        ;  ALL REGISTERS, STATUS ETC.  (3) TEST FOR THE
                        ;  NECESSARY TERMINATION.
                        ;***********************************************
                        ;-----------------------------------------------
                        ;       (1) PUT ALL REGISTERS, STATUS ETC INTO
                        ;           BUFFER.
                        ;-----------------------------------------------
07??    E5      BACK:   PUSH H          ;SAVE HL
07??    C5              PUSH B          ;SAVE BC
07??    F5              PUSH PSW        ;SAVE A & STATUS
07??    21              LXI  H,09F4     ;LOAD 1ST ADDR. OF BUFFER
07??    F4
07??    09
07??    C1              POP  B          ;FETCH PSW
07??    71              MOV  M,C        ;09F4_A
07??    23              INX  H          ;BUFFER ADDR. = 09F5
07??    70              MOV  M,B        ;09F5_STATUS
07??    23              INX  H          ;BUF. ADDR. = 09F6
```

```
                C1          POP  B       ;FETCH BC
                71          MOV  M,C     ;09F6_C
                23          INX  H       ;BUF. ADDR. = 09F7
                70          MOV  M,B     ;09F7_B
                23          INX  H       ;BUF. ADDR. = 09F8
                73          MOV  M,E     ;09F8_E
                23          INX  H       ;BUF. ADDR. = 09F9
                72          MOV  M,D     ;09F9_D
                23          INX  H       ;BUF. ADDR. = 09FA
                C1          POP  B       ;FETCH HL
                71          MOV  M,C     ;09FA_L
                23          INX  H       ;BUF. ADDR. =09FB
                70          MOV  M,B     ;09FB_H
                23          INX  H       ;BUF. ADDR. = 09FC
                C1 LSP:     POP  B       ;FETCH SP
                71          MOV  M,C     ;09FC_LSD OF SP
                23          INX  H       ;BUF. ADDR. = 09FD
                70          MOV  M,B     ;09FD_M.S.D. OF SP
                23          INX  H       ;BUF. ADDR. =09FE
                E5 LSPR:    PUSH H       ;INC. SP REGISTER & SAVE
                                         ;BUF. ADDR.
                21          LXI  H,0     ;CLEAR HL
                00
                00
                39          DAD  SP      ;LOAD SP REGISTER ONTO HL
                EB          XCHG         ;SAVE SP REGISTER
                E1          POP  H       ;RESTORE SP REGISTER
                73          MOV  M,E     ;09FE_LSD OF SP REG.
                23          INX  H       ;BUF. ADDR. =09FF
                72          MOV  M,D     ;09FF_M.S.D. OF SP REG.
                ;-----------------------------------------------
                ;  (2) O/P ALL REGISTERS, STSTUS, ETC.
                ;-----------------------------------------------
                3A OPRG:    LDA  0A0C
                0C
                0A
                FE          CPI  'N'
                4E
                CA          JZ   TSTM
                70
                0C
                CD          CALL CR      ;TYPE 'CR & LF'
                F9
                0F
                C3          JMP  0C00
                00
                0C
                00          NOP
                00          NOP
                00          NOP
                00          NOP
                00          NOP
                00          NOP
                00 .ORG 0C00
                2A          LHLD 0A0E    ;O/P
                0E
                0A
                CD          CALL OPSA    ;   PC
                7B
                0F
                CD          CALL TYSP
                DC
                0F
                2A          LHLD 0A04    ;O/P
                04
                0A
                CD          CALL OPSL    ;   1ST BYTE OF INSTR.
                8B
                0F
                CD          CALL TYSP
                DC
                0F
                3A          LDA  0A0D    ;FETCH BYTE COUNTER
                0D
                0A
                FE          CPI  02
                02
                CA          JZ   B2
                31
                0C
                FE          CPI  01
                01
                DC          CC   TYSP
                DC
                0F
                DC          CC   TYSP
                DC
                0F
                2A          LHLD 0A10    ;O/P 2ND BYTE
                10
                0A
                CC          CZ   OPSL
                8B
                0F
```

```
103 0C28  CD          CALL TYSP
    0C29  DC
    0C2A  0F
104 0C2B  CD          CALL TYSP
    0C2C  DC
    0C2D  0F
105 0C2E  C3          JMP  TBY
    0C2F  37
    0C30  0C
106 0C31  2A B2:      LHLD 0A10     ;O/P 2ND & 3RD
    0C32  10
    0C33  0A
107 0C34  CD          CALL OPSA     ;   BYTE IF COUNTER>1
    0C35  7B
    0C36  0F
108 0C37  CD TBY:     CALL TYSP
    0C38  DC
    0C39  0F
109 0C3A  2A          LHLD 09F4     ;O/P
    0C3B  F4
    0C3C  09
110 0C3D  CD          CALL OPSA     ;   PSW
    0C3E  7B
    0C3F  0F
111 0C40  CD          CALL TYSP
    0C41  DC
    0C42  0F
112 0C43  2A          LHLD 09F6     ;O/P
    0C44  F6
    0C45  09
113 0C46  CD          CALL OPSA     ;   BC
    0C47  7B
    0C48  0F
114 0C49  CD          CALL TYSP
    0C4A  DC
    0C4B  0F
115 0C4C  2A          LHLD 09F8     ;O/P
    0C4D  F8
    0C4E  09
116 0C4F  CD          CALL OPSA     ;   DE
    0C50  7B
    0C51  0F
117 0C52  CD          CALL TYSP
    0C53  DC
    0C54  0F
118 0C55  2A          LHLD 09FA     ;O/P
    0C56  FA
    0C57  09
119 0C58  CD          CALL OPSA     ;   HL
    0C59  7B
    0C5A  0F
120 0C5B  CD          CALL TYSP
    0C5C  DC
    0C5D  0F
121 0C5E  2A          LHLD 09FC     ;O/P
    0C5F  FC
    0C60  09
122 0C61  CD          CALL OPSA     ;   SP
    0C62  7B
    0C63  0F
123 0C64  CD          CALL TYSP
    0C65  DC
    0C66  0F
124 0C67  2A          LHLD 09FE     ;O/P
    0C68  FE
    0C69  09
125 0C6A  CD          CALL OPSA     ;   SP REG.
    0C6B  7B
    0C6C  0F
126 0C6D  CD          CALL TYSP
    0C6E  DC
    0C6F  0F
127               ;-----------------------------------------------
128               ;  (3) TEST FOR TERMINATION
129               ;-----------------------------------------------
130 0C70  2A TSTM:    LHLD 0A02     ;FETCH TERMINATE PC
    0C71  02
    0C72  0A
131 0C73  AF          XRA  A        ;CLEAR ACC.
132 0C74  32          STA  0A0D     ;CLEAR BYTE COUNTER
    0C75  0D
    0C76  0A
133 0C77  BC          CMP  H
134 0C78  C2          JNZ  TPC      ;IF TERM. PC NOT=0
    0C79  9A
    0C7A  0C
135 0C7B  BD          CMP  L
136 0C7C  C2          JNZ  TPC      ;IF TERM. PC NOT=0
    0C7D  9A
    0C7E  0C
137 0C7F  CD T9:      CALL SRIP2    ;READ
    0C80  5C
    0C81  01
```

```
130 0CB2    FE          CPI  0D
    0CB3    0D
139 0CB4    CA          JZ   ROUN     ;IF = 'CR'
    0CB5    15
    0CB6    06
140 0CB7    FE          CPI  'T'
    0CB8    54
141 0CB9    CA          JZ   TC       ;IF = 'T'
    0CBA    D6
    0CBB    05
142 0CBC    FE          CPI  'X'
    0CBD    58
143 0CBE    C2          JNZ  T9       ;IF NOT='X'
    0CBF    7F
    0C90    0C
144 0C91    CD          CALL ECHO
    0C92    6A
    0C93    0F
145 0C94    CD          CALL X        ;ENTER X COMMAND
    0C95    FE
    0C96    04
146 0C97    C3          JMP  ROUN
    0C98    15
    0C99    06
147 0C9A    EB TPC:     XCHG
148 0C9B    2A          LHLD 0A00     ;FETCH RECENT PC
    0C9C    00
    0C9D    0A
149 0C9E    7C          MOV  A,H      ;M.S.D. OF RECENT PC TO ACC.
150 0C9F    BA          CMP  D
151 0CA0    C2          JNZ  ROUN     ;IF MSD OF RECENT PC NOT=TERM. PC
    0CA1    15
    0CA2    06
152 0CA3    7D          MOV  A,L      ;LSD OF RECENT PC TO ACC.
153 0CA4    BB          CMP  E
154 0CA5    C2          JNZ  ROUN     ;IF TERM. PC NOT= RECENT PC
    0CA6    15
    0CA7    06
155 0CA8    AF          XRA  A
156 0CA9    32          STA  0A0C
    0CAA    0C
    0CAB    0A
157 0CAC    32          STA  0A02
    0CAD    02
    0CAE    0A
158 0CAF    32          STA  0A03
    0CB0    03
    0CB1    0A
159 0CB2    C3          JMP  ROUN     ;SET O/P AT LAST INSTR. CYCLE
    0CB3    15
    0CB4    06
```



```
  1             ;*************************************************
  2             ;
  3             ;            W - WRITE COMMAND
  4             ;       ( IMPLEMENTATION PROGRAM )
  5             ;
  6             ;       NAME: WRI
  7             ;       THIS ROUTINE ENTERED BY TYPING:-
  8             ;   >W START ADDRESS TERMINATE ADDRESS
  9             ;       THE CONTENT OF THE MEMORIES WOULD BE TYPE OUT
 10             ;  FROM THE START ADDRESS TO THE TERMINATE ADDRESS
 11             ;  IN THE FORM:-
 12             ; MMMM 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
 13             ;       WHERE: MMMM= ADDRESS OF FIRST DATA
 14             ;              00  = DATA
 15             ;
 16             ;*************************************************
 17 0CB5    CD WRI:     CALL TYSP     ;TYPE ' '
    0CB6    DC
    0CB7    0F
 18 0CB8    CD          CALL INAD     ;I/P START ADDR.
    0CB9    03
    0CBA    0F
 19 0CBB    EB          XCHG          ;SHIFT START ADDR. TO HL
 20 0CBC    CD          CALL INAT     ;I/P FIN.ADDR. & CHECK IF ACCEPTED
    0CBD    9C
    0CBE    0F
 21 0CBF    42          MOV  B,D      ;SAVE FINISH ADDR. IN
 22 0CC0    4B          MOV  C,E      ;          BC
 23 0CC1    16 NXLI:    MVI  D,0      ;CLEAR D FOR NO. OF DATA O/P COUNT
    0CC2    00
 24 0CC3    CD          CALL OPSA     ;O/P START/FIRST ADDR.
    0CC4    7B
    0CC5    0F
 25 0CC6    C5 HFLI:    PUSH B        ;SAVE BC
 26 0CC7    CD          CALL TYSP     ;TYPE ' '
    0CC8    DC
    0CC9    0F
 27 0CCA    C1          POP  B        ;RESTORE BC
 28 0CCB    5E          MOV  E,M      ;FETCH MEMORY CONTENT
 29 0CCC    23          INX  H        ;INC. ADDR.
```

```
30 0CCD    EB          XCHG          ;SAVE ADDR. & SHIFT DATA TO L
31 0CCE    CD          CALL OPSL     ;O/P MEMORY CONTENT
   0CCF    8B
   0CD0    0F
32 0CD1    EB          XCHG          ;RESTORE ADDR.
33 0CD2    14          INR  D         ;INC. O/P COUNT
34 0CD3    7A          MOV  A,D
35 0CD4    FE          CPI  8
   0CD5    08
36 0CD6    C2          JNZ  W1       ;IF O/P COUNT NOT EQ. 8
   0CD7    DC
   0CD8    0C
37 0CD9    CD          CALL TYSP     ;TYPE ' '
   0CDA    DC
   0CDB    0F
38 0CDC    E5 W1:      PUSH H
39 0CDD    7C          MOV  A,H
40 0CDE    2F          CMA
41 0CDF    67          MOV  H,A
42 0CE0    7D          MOV  A,L
43 0CE1    2F          CMA
44 0CE2    6F          MOV  L,A
45 0CE3    23          INX  H
46 0CE4    09          DAD  B        ;BC-HL
47 0CE5    E1          POP  H
48 0CE6    D2          JNC  TREN     ;*STOP* IF -VE
   0CE7    9D
   0CE8    04
49 0CE9    7A          MOV  A,D      ;FETCH O/P COUNT
50 0CEA    FE          CPI  10
   0CEB    10
51 0CEC    C2          JNZ  NFLI     ;IF NOT NEXT LINE
   0CED    C6
   0CEE    0C
52 0CEF    CD          CALL CR       ;TYPE 'CR & LF'
   0CF0    F9
   0CF1    0F
53 0CF2    C3          JMP  NXLI     ;TYPE NEXT LINE
   0CF3    C1
   0CF4    0C
54              ;*****************************************************
55              ;
56              ;              M - MODIFY COMMAND
57              ;        ( IMPLEMENTION PROGRAM )
58              ;
59              ;       NAME: MODY
60              ;       THIS ROUTINE IS INTERED BY TYPING:-
61              ;       >M ADDRESS
62              ;       IT WILL TYPE OUT:-
63              ;       MMMM DD (NEW DATA)
64              ;       WHERE: MMMM= MOMERY ADDRESS
65              ;              DD  = DATA CONTENT IN THAT ADDRESS
66              ;       (NEW DATA) IS TYPED IN FOLLOWED BY A CR.
67              ;       IF CR OR ^ IS TYPED IN INSTEAD OF (NEW DATA)
68              ;  THE PROGRAM WILL KEEP THE ORIGINAL DATA OR KEEP
69              ;  THE ORIGINAL DATA & DEC. MEMORY ADDR. RESPECTIVELY.
70              ;       IF A NEW DATA IS WRONGLY TYPED, IT CAN BE
71              ;  RETYPE BY TYPING 'RUB OUT' AND TYPE AGAIN.
72              ;
73              ;*****************************************************
74 0CF5    CD MODY:    CALL INAT     ;I/P START ADDR. & CHECK FOR ACCEP.
   0CF6    9C
   0CF7    0F
75 0CF8    EB          XCHG          ;STORE START ADDR. IN HL
76 0CF9    CD TYAD:    CALL OPSA     ;O/P ADDR.
   0CFA    7B
   0CFB    0F
77 0CFC    CD          CALL TYSP     ;TYPE ' '
   0CFD    DC
   0CFE    0F
78 0CFF    5E          MOV  E,M      ;FETCH DATA
79 0D00    EB          XCHG          ;SAVE ADDR. & DATA TO L
80 0D01    CD          CALL OPSL     ;O/P DATA
   0D02    8B
   0D03    0F
81 0D04    EB          XCHG          ;RESTORE ADDR.
82 0D05    CD          CALL TYSP     ;TYPE ' '
   0D06    DC
   0D07    0F
83 0D08    CD INDT:    CALL RASC     ;I/P NEW DATA
   0D09    50
   0D0A    0F
84 0D0B    DA          JC   M1       ;IF NO NEW DATA
   0D0C    1A
   0D0D    0D
85 0D0E    CD          CALL INAD3    ;I/P NEW DATA
   0D0F    27
   0D10    0F
86 0D11    DA          JC   M1       ;IF NOT HEX.
   0D12    1A
   0D13    0D
```

```
 87 0D14    CD          CALL SKIP2   ;READ
    0D15    5C
    0D16    01
 88 0D17    CD          CALL ECHO
    0D18    6A
    0D19    0F
 89 0D1A    FE M1:      CPI  0D
    0D1B    0D
 90 0D1C    C2          JNZ  M2      ;IF NOT = CR
    0D1D    24
    0D1E    0D
 91 0D1F    73          MOV  M,E     ;REPLACE OLD WITH NEW DATA
 92 0D20    23          INX  H       ;INC. ADDR.
 93 0D21    C3          JMP  TYAD    ;MODY NEXT ADDR.
    0D22    F9
    0D23    0C
 94 0D24    FE M2:      CPI  5E
    0D25    5E
 95 0D26    C2          JNZ  INDT    ;IF NOT='^' READ AGAIN
    0D27    08
    0D28    0D
 96 0D29    CD          CALL CR      ;TYPE 'CR & LF'
    0D2A    F9
    0D2B    0F
 97 0D2C    73          MOV  M,E     ;REPLACE OLD WITH NEW DATA
 98 0D2D    2B          DCX  H       ;DEC. ADDR.
 99 0D2E    C3          JMP  TYAD    ;MODY. ADDR. - 1
    0D2F    F9
    0D30    0C
100             ;***********************************************
101             ;
102             ;              E - ENTER COMMAND
103             ;        ( IMPLEMENTATION PROGRAM )
104             ;
105             ;       NAME: ENTER
106             ;       THIS ROUTINE CAN BE ENTERED BY TYPING:-
107             ;       >E START ADDRESS
108             ;       THE PROGRAM WOULD TYPE OUT:
109             ;       START ADDRESS (DATA) (DATA) ..... 'CR'
110             ;       THE PROGRAM WOULD THEN TYPE OUT THE STARTING
111             ;  ADDRESS OF ANOTHER ROW. YOU CAN TYPE IN DATA AGAIN
112             ;  AS IN ABOVE. TO END THE INPUT, USE ^C.
113             ;       NOTE: (DATA) ARE DATAS THAT YOU TYPE IN.
114             ;
115             ;       CALLS: INAT, OPSA
116             ;
117             ;***********************************************
118 0D31    CD ENTER:   CALL INAT    ;I/P START ADDR. & TEST WHETHER
    0D32    9C
    0D33    0F
119                                  ;          ACCEPTED.
120 0D34    EB          XCHG         ;LOAD ADDR. ONTO HL
121 0D35    CD STRW:    CALL OPSA    ;O/P START ADDR.
    0D36    7B
    0D37    0F
122 0D38    CD NXDA:    CALL TYSP    ;TYPE ' '
    0D39    DC
    0D3A    0F
123 0D3B    CD E1:      CALL RASC    ;I/P 1ST DIGIT OF DATA
    0D3C    50
    0D3D    0F
124 0D3E    D2          JNC  E3      ;IF HEX.
    0D3F    49
    0D40    0D
125 0D41    FE E2:      CPI  0D
    0D42    0D
126 0D43    C2          JNZ  E1      ;IF NOT 'CR' READ AGAIN
    0D44    3B
    0D45    0D
127 0D46    C3          JMP  STRW    ;START NEW ROW
    0D47    35
    0D48    0D
128 0D49    CD E3:      CALL INAD3   ;I/P 2ND DIGIT & PUT BOTH ONTO E
    0D4A    27
    0D4B    0F
129 0D4C    73          MOV  M,E     ;STORE DATA IN MEMORY
130 0D4D    23          INX  H       ;INC. MEMORY ADDR.
131 0D4E    DA          JC   E2      ;IF NOT HEX. GO TO ANOTHER ROW
    0D4F    41
    0D50    0D
132 0D51    C3          JMP  NXDA    ;I/P NEXT DATA
    0D52    38
    0D53    0D
```

```
                ;*************************************************
                ;
                ;              C - COPY
                ;         ( IMPLEMENTATION PROGRAM )
                ;
                ;       THIS ROUTINE CAN BE ENTERED BY:-
                ;  >C START ADDRESS TERMINATE ADDRESS START ADDRESS
                ;       OF SOURCE      OF SOURCE      OF DESTINATION
                ;
                ;       THIS ROUTINE COPIES FROM ONE MEMORY LOCATION
                ;  TO ANOTHER.
                ;
                ;*************************************************
                ;I/P ADDRESSES ETC.
0D54  CD  COPY: CALL TYSP       ;TYPE ' '
0D55  DC
0D56  0F
0D57  CD        CALL INAD       ;I/P START ADDR. OF SOURCE
0D58  03
0D59  0F
0D5A  EB        XCHG            ;START ADDR. OF SOURCE
                                ;ONTO HL
0D5B  CD        CALL TYSP       ;TYPE ' '
0D5C  DC
0D5D  0F
0D5E  CD        CALL INAD       ;I/P TERM. ADDR. OF SOURCE
0D5F  03
0D60  0F
0D61  42        MOV  B,D        ;STORE TERM. ADDR. OF SOURCE
0D62  4B        MOV  C,E        ;       IN BC
0D63  CD        CALL INAT       ;I/P START ADDR. OF DEST. &
0D64  9C
0D65  0F
                                ;    CHECK FOR ACCEPTANCE.
0D66  7A        MOV  A,D        ;M.S.D. OF DEST. TO ACC.
0D67  BC        CMP  H          ;COMP. DEST. WITH SOURCE
0D68  CA        JZ   C1         ;IF M.S.D. EQ.
0D69  71
0D6A  0D
0D6B  DA        JC   CPST       ;IF M.S.D. OF DEST. < SOURCE
0D6C  79
0D6D  0D
0D6E  C3        JMP  CPTM       ;IF M.S.D. OF DEST. > SOURCE
0D6F  92
0D70  0D
0D71  7B  C1:   MOV  A,E        ;L.S.D. OF DEST. TO ACC.
0D72  BD        CMP  L          ;COMP. DEST. WITH SOURCE
0D73  CA        JZ   TREN       ;IF EQ. JMP TO *STOP*
0D74  9D
0D75  04
0D76  D2        JNC  CPTM       ;IF DEST. > SOURCE
0D77  92
0D78  0D
                ;-------------------------------------------------
                ;COPY FROM START ADDR. OF SOURCE IF START ADDR. OF
                ;DEST. < SOURCE.
                ;-------------------------------------------------
0D79  7E  CPST: MOV  A,M        ;FETCH DATA STARTING FROM
                                ;       START ADDR.
0D7A  23        INX  H          ;INC. SOURCE ADDR.
0D7B  EB        XCHG
0D7C  77        MOV  M,A        ;MOVE DATA TO DESTINATION
0D7D  23        INX  H          ;INC. DEST. ADDR.
0D7E  EB        XCHG
0D7F  7C        MOV  A,H        ;FETCH M.S.D. OF SOURCE ADDR.
0D80  B8        CMP  B
0D81  DA        JC   CPST       ;M.S.D. OF SOURCE < TERM. ADDR.
0D82  79
0D83  0D
0D84  C2        JNZ  CRST       ;M.S.D. OF SOURCE > TERM. ADDR.
0D85  8C
0D86  0D
0D87  79        MOV  A,C        ;FETCH L.S.D. OF TERM ADDR.
0D88  BD        CMP  L
0D89  D2        JNC  CPST       ;L.S.D. OF TERM.>=SOURCE ADDR.
0D8A  79
0D8B  0D
0D8C  CD  CRST: CALL CR         ;TYPE 'CR & LF'
0D8D  F9
0D8E  0F
0D8F  C3        JMP  TREN       ;JMP TO *STOP*
0D90  9D
0D91  04
```

```
                        ;-------------------------------------------------
                        ;COPY FROM TERM. ADDR. OF SOURCE IF START ADDR. OF
                        ;DEST. > SOURCE.
                        ;-------------------------------------------------
                        ;THE FOLLOW SECTION SUBTRACT TERMINATE ADDRESS OF
                        ;SOURCE BY STARTING ADDRESS OF SOURCE.  THE RESULT
                        ;STORE IN DE.  ALSO, MOVE TERMINATE ADDRESS OF SOURCE
                        ;TO HL AND STARTING ADDRESS OF DESTINATION
                        ;TO BC.
 63 0D92     79 CPTM:   MOV  A,C      ;L.S.D. TERM. ADDR.
 64 0D93     95         SUB  L        ;         - START ADDR.
 65 0D94     69         MOV  L,C      ;L_TERM. ADDR.
 66 0D95     4B         MOV  C,E      ;C_START OF DEST. ADDR.
 67 0D96     5F         MOV  E,A      ;E_TERM-START
 68 0D97     78         MOV  A,B      ;M.S.D. TERM ADDR.
 69 0D98     9C         SBB  H        ;       - START ADDR.
 70 0D99     60         MOV  H,B      ;H_TERM. ADDR.
 71 0D9A     42         MOV  B,D      ;B_START OF DEST. ADDR.
 72 0D9B     57         MOV  D,A      ;D_TERM-START
 73                     ;ADD DIFFERENCE OF TERMINATE ADDRESS OF SOURCE AND
 74                     ;STARTING ADDRESS OF SOURCE TO STARTING ADDRESS
 75                     ;OF DESTINATION AND PLACE THE SUM IN HL.  THE
 76                     ;TERMINATE ADDRESS OF SOURCE IS BEING MOVE TO DE.
 77 0D9C     EB         XCHG          ;HL_DIFF., DE_TERM. ADDR.
 78 0D9D     09         DAD  B        ;DIFF. + START ADDR. OF DEST.
 79 0D9E     EB CRBK:   XCHG          ;REORGANISE DEST. & SOURCE
 80 0D9F     7E         MOV  A,M      ;FETCH DATA
 81 0DA0     2B         DCX  H        ;DEC. ADDR. OF SOURCE
 82 0DA1     EB         XCHG
 83 0DA2     77         MOV  M,A      ;REPLACE DATA
 84 0DA3     2B         DCX  H        ;DEC. ADDR. OF DEST.
 85 0DA4     78         MOV  A,B      ;
 86 0DA5     BC         CMP  H
 87 0DA6     DA         JC   CRBK     ;IF START < RECENT ADDR.
    0DA7     9E
    0DA8     0D
 88 0DA9     C2         JNZ  CRST     ;IF M.S.D. OF START > RECENT ADDR.
    0DAA     BC
    0DAB     0D
 89 0DAC     7D         MOV  A,L
 90 0DAD     B9         CMP  C
 91 0DAE     D2         JNC  CRBK     ;IF RECENT ADDR.>=START ADDR.
    0DAF     9E
    0DB0     0D
 92 0DB1     C3         JMP  CRST     ;TYPE 'CR''LF' THEN *STOP*
    0DB2     BC
    0DB3     0D
TELETYPE MONITOR - 8080              CALOS-80 V2.12 00/00/70 PAGE   20

  1                     ;*************************************************
  2                     ;
  3                     ;              D - DUMP COMMAND
  4                     ;        ( IMPLEMENTATION PROGRAM )
  5                     ;
  6                     ;       THIS ROUTINE IS ENTERED BY TYPING:-
  7                     ;  >D START ADDRESS TERMINATE ADDRESS
  8                     ;       IT WILL DUMP DATA OUT IN THE FORM:-
  9                     ;  :NNMMMMOODTDTDTDTDTDTDTDTDTDTDTDTDTDTDTCS
 10                     ;       WHERE: NN   = NUMBER OF HEX. DATA WORDS
 11                     ;              MMMM= ADDRESS OF 1ST DATA WORD
 12                     ;              DT   = DATA
 13                     ;              CS   = CHECK SUM (NEGATED SUM OF ALL
 14                     ;                     BYTES)
 15                     ;
 16                     ;       CALLS: SROP2,ECHO,TECH,INAT,INAD,TYSP,OPSA
 17                     ;
 18                     ;*************************************************
 19 0DB4     CD DUMP:   CALL TYSP     ;TYPE ' '
    0DB5     DC
    0DB6     0F
 20 0DB7     CD         CALL INAD     ;I/P START ADDR.
    0DB8     03
    0DB9     0F
 21 0DBA     EB         XCHG          ;HL_START ADDR.
 22 0DBB     CD         CALL INAT     ;I/P TERM. ADDR. & CHECK
    0DBC     9C
    0DBD     0F
 23                                   ;    WHETHER ACCEPTED
 24 0DBE     CD         CALL ECHO     ;TYPE 'CR' & 'LF'
    0DBF     6A
    0DC0     0F
 25 0DC1     0E         MVI  C,0      ;TYPE 18 'NUL' (ADVANCE
    0DC2     00
 26 0DC3     CD         CALL TECH     ;     PAPER TAPE)
    0DC4     E4
    0DC5     0F
 27                     ;SECTION FOR LISTING
 28 0DC6     0E DLIST:  MVI  C,':'    ;TYPE
    0DC7     3A
 29 0DC8     CD         CALL SROP2    ;      ':'
    0DC9     17
    0DCA     01
 30 0DCB     E5         PUSH H        ;SAVE HL (START ADDR.)
 31 0DCC     D5         PUSH D        ;     DE (TERMINATE ADDR.)
                                A-10
```

```
32                      ;DE=(HL-1) (TERM.-(START ADDR.-1))
33 0DC0  7C             MOV  A,H
34 0DC1  2F             CMA
35 0DC2  67             MOV  H,A
36 0DC3  7D             MOV  A,L
37 0DC4  2F             CMA
38 0DC5  6F             MOV  L,A
39 0DC6  23             INX  H
40 0DC7  23             INX  H
41 0DC8  19             DAD  D
42 0DC9  7C             MOV  A,H      ;IF M.S. DIGITS
43 0DCA  FE             CPI  0        ;   NOT
   0DCB  00
44 0DCC  C2             JNZ  SXTN     ;   0
   0DCD  FB
   0DCE  0D
45 0DCF  7D             MOV  A,L      ;IF L.S. DIGITS
46 0DD0  FE             CPI  0        ;    NOT
   0DD1  00
47 0DD2  C2             JNZ  TSX      ;    0
   0DD3  EF
   0DD4  0D
48 0DD5  0E             MVI  C,'0'    ;TYPE
   0DD6  30
49 0DD7  CD             CALL TECH     ;      '0'S
   0DD8  E4
   0DD9  0F
50 0DDA  0E             MVI  C,0      ;ADVANCE PAPER
   0DDB  00
51 0DDC  CD             CALL TECH     ;        TAPE
   0DDD  E4
   0DDE  0F
52 0DDF  C3             JMP  TREN     ;*STOP*
   0DE0  9D
   0DE1  04
53 0DE2  FE TSX:        CPI  10
   0DE3  10
54 0DE4  D2             JNC  SXTN     ;IF >= 16
   0DE5  FB
   0DE6  0D
55 0DE7  45             MOV  B,L      ;SET COUNTER
56 0DE8  CD             CALL OPSL     ;O/P NO. OF DATA BYTES
   0DE9  BB
   0DEA  0F
57 0DEB  C3             JMP  ACKS
   0DEC  01
   0DED  0E
58 0DEE  06 SXTN:       MVI  B,10     ;SET COUNTER
   0DEF  10
59 0DF0  68             MOV  L,B      ;O/P 10
60 0DF1  CD             CALL OPSL     ;    (NO. OF DATA BYTES)
   0DF2  BB
   0DF3  0F
61 0DF4  48 ACKS:       MOV  C,B      ;ADD TO CHECK SUM
62 0DF5  D1             POP  D        ;RESTORE TERMINATE ADDR.
63 0DF6  E1             POP  H        ;RESTORE START ADDR.
64 0DF7  CD             CALL OPSA     ;O/P START ADDR.
   0DF8  7B
   0DF9  0F
65 0DFA  79             MOV  A,C      ;ADD
66 0DFB  84             ADD  H        ;    TO
67 0DFC  85             ADD  L        ;    CHECK
68 0DFD  4F             MOV  C,A      ;    SUM
69 0DFE  3E             MVI  A,'0'    ;TYPE
   0DFF  30
70 0E00  CD             CALL ECHO     ;     TWO
   0E01  6A
   0E02  0F
71 0E03  CD             CALL ECHO     ;     '0'S
   0E04  6A
   0E05  0F
72 0E06  7E DMDA:       MOV  A,M      ;FETCH DATA
73 0E07  E5             PUSH H        ;O/P DATA
74 0E08  6F             MOV  L,A      ;    IN
75 0E09  CD             CALL OPSL     ;    HEX
   0E0A  BB
   0E0B  0F
76 0E0C  E1             POP  H        ;
77 0E0D  7E             MOV  A,M      ;ADD TO
78 0E0E  81             ADD  C        ;    CHECK
79 0E0F  4F             MOV  C,A      ;    SUM
80 0E10  05             DCR  B        ;DEC. COUNTER
81 0E11  23             INX  H        ;INC. ADDR.
82 0E12  AF             XRA  A
83 0E13  B8             CMP  B
84 0E14  C2             JNZ  DMDA     ;IF COUNTER NOT = 0
   0E15  06
   0E16  0E
85 0E17  E5             PUSH H
86 0E18  79             MOV  A,C      ;TWO'S COMPLEMENT
87 0E19  2F             CMA           ;  CHECK
88 0E1A  3C             INR  A        ;  SUM
```

```
                MOV  L,A        ;O/P CHECK SUM
                CALL OPSL       ;    IN
                POP  H          ;    HEX
                CALL CR         ;TYPE 'CR & LF'
                JMP  DLIST
TELETYPE MONITOR - 8080                 CALOS-80 V2.12 00/00/70 PAGE   21

                ;***************************************************
                ;
                ;               L - LOAD
                ;         ( IMPLEMENTATION PROGRAM )
                ;
                ;       TO ENTER THIS MODE TYPE:-
                ;       >L CR
                ;       THIS WILL LOAD FROM PAPER TAPE ACCORDING
                ;  TO START ADDRESS ON THE TAPE.
                ;       >L ADDRESS 1 CR
                ;       THIS WILL LOAD FROM PAPER TAPE STARTING
                ;  FROM ADDRESS 1.
                ;       >L ADDRESS 1 ADDRESS 2 CR
                ;       THIS WILL LOAD FROM PAPER TAPE STARTING
                ;  FROM ADDRESS 1 AND TERMINATE AT ADDRESS 2.
                ;
                ;       CALLS: RASC, SRIP2, ECHO, INAD
                ;       ADDRESS 1 IN HL, ADDRESS 2 IN BC.
                ;
                ;***************************************************
0E33 CD LOAD:   CALL TYSP       ;TYPE ' '
0E34 DC
0E35 0F
0E36 01         LXI  B,0        ;CLEAR BC
0E37 00
0E38 00
0E39 CD L1:     CALL RASC       ;READ ONE DIGIT
0E3A 50
0E3B 0F
0E3C D2         JNC  RDST       ;IF HEX. DIGIT(FOR ST. ADDR.)
0E3D 47
0E3E 0E
0E3F FE         CPI  0D
0E40 0D
0E41 C2         JNZ  L1         ;IF NOT 'CR', READ AGAIN
0E42 39
0E43 0E
0E44 C3         JMP  STP        ;START LOADING FROM TAPE
0E45 70
0E46 0E
0E47 CD RDST:   CALL INAD1      ;READ START ADDR.
0E48 09
0E49 0F
0E4A EB         XCHG            ;START ADDR. TO HL
0E4B 16         MVI  D,0F0      ;SET ADDR. STATUS
0E4C F0
0E4D CD         CALL TYSP       ;TYPE ' '
0E4E DC
0E4F 0F
0E50 CD L2:     CALL RASC       ;READ ONE DIGIT
0E51 50
0E52 0F
0E53 D2         JNC  L3         ;IF HEX. DIGIT (TERM. ADDR.)
0E54 5E
0E55 0E
0E56 FE         CPI  0D         ;IF
0E57 0D
0E58 C2         JNZ  L2         ;   NOT 'CR'
0E59 50
0E5A 0E
0E5B C3         JMP  STP        ;GO TO START TAPE
0E5C 70
0E5D 0E
0E5E CD L3:     CALL INAD1      ;READ TERM. ADDR.
0E5F 09
0E60 0F
0E61 42         MOV  B,D        ;TERM. ADDR.
0E62 4B         MOV  C,E        ;     TO BC
0E63 16         MVI  D,0FF      ;SET ADDR. STATUS
0E64 FF
0E65 DA         JC   L5         ;IF NOT HEX. DIGIT
0E66 6B
0E67 0E
0E68 CD L4:     CALL SRIP2      ;READ
0E69 5C
0E6A 01
0E6B FE L5:     CPI  0D         ;IF
0E6C 0D
0E6D C2         JNZ  L4         ;   NOT CR
0E6E 68
0E6F 0E
```

```
45                      ;START LOADING
46 0E70    C5 STP:      PUSH B          ;SAVE TERMINATION ADDR.
47 0E71    CD BGLN:     CALL ECHO       ;ECHO TYPE
   0E72    6A
   0E73    0F
48 0E74    3A L7:       LDA  0BFC
   0E75    FC
   0E76    0B
49 0E77    E6           ANI  01
   0E78    01
50 0E79    C2           JNZ  L7
   0E7A    74
   0E7B    0E
51                      ;ENABLE PAPER TAPE READER
52 0E7C    3A           LDA  0BFC
   0E7D    FC
   0E7E    0B
53 0E7F    F6           ORI  2
   0E80    02
54 0E81    32           STA  0BFC
   0E82    FC
   0E83    0B
55 0E84    32           STA  8011
   0E85    11
   0E86    80
56 0E87    CD L8:       CALL SRIP2      ;WAIT FOR
   0E88    5C
   0E89    01
57 0E8A    FE           CPI  ':'        ;     CHARACTER
   0E8B    3A
58 0E8C    C2           JNZ  L8         ;     ':'
   0E8D    87
   0E8E    0E
59 0E8F    CD           CALL ECHO
   0E90    6A
   0E91    0F
60 0E92    CD           CALL INAD2      ;I/P NO. OF DATA BYTE
   0E93    1C
   0E94    0F
61 0E95    43           MOV  B,E        ;LOAD COUNTER
62 0E96    4B           MOV  C,E        ;LOAD CHECK SUM
63 0E97    FE           CPI  0
   0E98    00
64 0E99    C2           JNZ  LDD        ;IF NOT 0 GO TO LOAD DATA
   0E9A    A2
   0E9B    0E
65                      ;DISABLE READER AND STOP
66 0E9C    CD L6:       CALL DRD        ;DISABLE TAPE READER
   0E9D    C3
   0E9E    0F
67 0E9F    C3           JMP  TREN       ;*STOP*
   0EA0    9D
   0EA1    04
68                      ;LOAD ADDR. AND DATA
69 0EA2    D5 LDD:      PUSH D
70 0EA3    7A           MOV  A,D
71 0EA4    FE           CPI  0F0
   0EA5    F0
72 0EA6    CA           JZ   WSOY       ;IF START ADDR. BEEN LOADED ONLY
   0EA7    B5
   0EA8    0E
73 0EA9    FE           CPI  0FF
   0EAA    FF
74 0EAB    CA           JZ   WSAT       ;IF START & TERM. ADDR. WERE
   0EAC    BC
   0EAD    0E
75                                      ;LOADED
76 0EAE    CD           CALL IACS       ;I/P INITIAL ADDR. & ADD TO
   0EAF    F1
   0EB0    0F
77                                      ;CHECK SUM
78 0EB1    EB           XCHG            ;STORE INITIAL ADDR.
79 0EB2    C3           JMP  PZOS
   0EB3    B8
   0EB4    0E
80 0EB5    CD WSOY:     CALL IACS       ;I/P & IGNORE INITIAL ADDR.
   0EB6    F1
   0EB7    0F
81 0EB8    D1 PZOS:     POP  D          ;RESTORE ADDR. STATUS
82 0EB9    C3           JMP  ZOS
   0EBA    DC
   0EBB    0E
83 0EBC    CD WSAT:     CALL IACS       ;I/P & IGNORE INITIAL ADDR.
   0EBD    F1
   0EBE    0F
84 0EBF    D1           POP  D          ;RESTORE ADDR. STATUS
85 0EC0    7A           MOV  A,D        ;SAVE ADDR. STATUS
86 0EC1    D1           POP  D          ;FETCH
87 0EC2    D5           PUSH D          ;     TERMINATE ADDR.
88 0EC3    E5           PUSH H          ;SAVE START ADDR.
89 0EC4    F5           PUSH PSW
```

```
 90 0EC3    7C          MOV  A,H      ;1'S
 91 0EC4    2F          CMA           ;
 92 0EC5    67          MOV  H,A      ;    COMPLEMENT
 93 0EC6    7D          MOV  A,L      ;    OF
 94 0EC7    2F          CMA           ;
 95 0EC8    6F          MOV  L,A      ;    HL
 96 0EC9    19          DAD  D        ;DE-HL+1
 97 0ECC    F1          POP  PSW
 98 0ECD    57          MOV  D,A      ;RESTORE ADDR. STATUS
 99 0ECE    AF          XRA  A
100 0ECF    BC          CMP  H
101 0ED0    C2          JNZ  HZOS     ;IF DE-HL+1>16
    0ED1    DB
    0ED2    0E
102 0ED3    7D          MOV  A,L
103 0ED4    B8          CMP  B
104 0ED5    D2          JNC  HZOS     ;IF DE-HL+1 NOT < B
    0ED6    DB
    0ED7    0E
105 0ED8    F6          ORI  80       ;SET TERM. FLAG
    0ED9    80
106 0EDA    47          MOV  B,A
107 0EDB    E1 HZOS:    POP  H        ;RESTORE START ADDR.
108 0EDC    CD ZOS:     CALL INAD2    ;I/P 2 'O'S
    0EDD    1C
    0EDE    0F
109 0EDF    CD IPD:     CALL INAD2    ;I/P ONE BYTE OF DATA
    0EE0    1C
    0EE1    0F
110 0EE2    73          MOV  M,E      ;MEMORY_DATA
111 0EE3    79          MOV  A,C      ;ADD
112 0EE4    83          ADD  E        ;    TO
113 0EE5    4F          MOV  C,A      ;    CHECK SUM
114 0EE6    05          DCR  B        ;DECREMENT COUNTER
115 0EE7    23          INX  H        ;INC. MEMORY ADDR.
116 0EE8    78          MOV  A,B
117 0EE9    FE          CPI  80
    0EEA    80
118 0EEB    CA          JZ   L6       ;IF TERM. FLAG SET
    0EEC    9C
    0EED    0E
119 0EEE    FE          CPI  0
    0EEF    00
120 0EF0    C2          JNZ  IPD      ;IF COUNTER NOT 0
    0EF1    DF
    0EF2    0E
121 0EF3    CD          CALL INAD2    ;READ CHECK SUM
    0EF4    1C
    0EF5    0F
122 0EF6    79          MOV  A,C
123 0EF7    83          ADD  E
124 0EF8    C4          CNZ  ER2      ;TYPE'ERROR' IF E NOT=2'S COMP. C
    0EF9    AB
    0EFA    01
125 0EFB    CD          CALL DRD      ;DISABLE PAPER TAPE READER
    0EFC    C3
    0EFD    0F
126 0EFE    3E          MVI  A,0D     ;O/P 'CR'
    0EFF    0D
127 0F00    C3          JMP  BGLN
    0F01    71
    0F02    0E
```



```
  1             ;***********************************************
  2             ;       (1)     UTILITY SUBROUTINE
  3             ;       NAME    : INAD
  4             ;       FUNCTION: FOR INPUTING A 4 HEX.DIGIT ADDR.
  5             ;                 ONTO REGISTER DE.
  6             ;       INPUTS  : NONE
  7             ;       OUTPUTS : 4 HEX. DIGITS IN REGISTER DE OR
  8             ;                 THE LAST DIGIT STORED IN ACCUMULATOR
  9             ;                 & CY FLAG SET.
 10             ;       CALLS   : RASC, SDRE
 11             ;       DESTROYS: ALL STATUS,A, D, E
 12             ;       DESCRIPTION: THIS ROUTINE INPUTS AND CONVERTS
 13             ;                    FROM ASCIII TO HEX. A 4 HEX. DIGIT
 14             ;                    NUMBER ONTO DE.
 15             ;                    IF ANYTHING OTHER THAT HEX.IS I/P
 16             ;                    THE NUMBER WOULD TERMINATE AND THE
 17             ;                    LAST CHARACTER INPUTTED IS RE-
 18             ;                    SERVED IN A AND CY FLAG SET.
 19             ;                    THERE ARE OTHER ENTRY POINTS TO
 20             ;                    THIS ROUTINE:-
 21             ;                    INAD1: FOR AVOIDING THE INITIAL
 22             ;                           LOOP.
 23             ;                    INAD2: FOR I/P 2 HEX. DIGITS ONLY
 24             ;                           (INTO E).
 25             ;                    INAD3: AS IN INAD2 BUT AVOIDING
 26             ;                           THE INITIAL LOOP.
 27             ;***********************************************
 28 0F03    CD INAD:    CALL RASC     ;I/P FIRST DIGIT & ECHO
    0F04    50
    0F05    0F
```

```
0F06  DA          JC   INAD     ;IF NOT HEX.
0F07  03
0F08  0F
0F09  07 INAD1:   RLC           ;LOAD
0F0A  07          RLC           ;      ONTO
0F0B  07          RLC           ;           FIRST
0F0C  07          RLC           ;      FOUR BIT
0F0D  57          MOV  D,A      ;      OF D
0F0E  CD          CALL RASC     ;I/P 2ND DIGIT
0F0F  50
0F10  0F
0F11  D2          JNC  IN1      ;IF HEX.
0F12  1A
0F13  0F
0F14  5A          MOV  E,D      ;COPY D TO E
0F15  16          MVI  D,0      ;      AND CLEAR D
0F16  00
0F17  C3          JMP  SHDE
0F18  37
0F19  0F
0F1A  B2 IN1:     ORA  D        ;LOAD HEX. ONTO 2ND
0F1B  57          MOV  D,A      ;      FOUR BIT OF D
0F1C  CD INAD2:   CALL RASC     ;I/P 3RD DIGIT
0F1D  50
0F1E  0F
0F1F  D2          JNC  INAD3    ;IF HEX.
0F20  27
0F21  0F
0F22  5A          MOV  E,D      ;COPY D TO E
0F23  16          MVI  D,0      ;      AND CLEAR D
0F24  00
0F25  37          STC           ;SET CY
0F26  C9          RET
0F27  07 INAD3:   RLC           ;LOAD
0F28  07          RLC           ;      ONTO
0F29  07          RLC           ;      FIRST
0F2A  07          RLC           ;      4 BIT
0F2B  5F          MOV  E,A      ;      OF E
0F2C  CD          CALL RASC     ;I/P 4TH DIGIT
0F2D  50
0F2E  0F
0F2F  DA          JC   SHDE     ;IF NOT HEX
0F30  37
0F31  0F
0F32  B3          ORA  E        ;LOAD ONTO 2ND 4 BIT
0F33  5F          MOV  E,A      ;      OF E
0F34  37          STC           ;CLEAR
0F35  3F          CMC           ;      CY
0F36  C9          RET
0F37  F5 SHDE:    PUSH PSW      ;
0F38  CD          CALL SDER     ;SHIFT
0F39  47
0F3A  0F
0F3B  CD          CALL SDER     ;      DE
0F3C  47
0F3D  0F
0F3E  CD          CALL SDER     ;      RIGHT
0F3F  47
0F40  0F
0F41  CD          CALL SDER     ;      4 BIT
0F42  47
0F43  0F
0F44  F1          POP  PSW
0F45  37          STC           ;SET CY
0F46  C9          RET
                  ;*************************************************
                  ;       (2)     UTILITY SUBROUTINE
                  ;       NAME    : SDER
                  ;       FUNCTION: TO SHIFT DE RIGHT
                  ;       INPUTS  : D, E
                  ;       OUTPUTS : D, E, STATUS, A
                  ;       CALLS   : NONE
                  ;       DESTROYS: ACC. & STATUS
                  ;       DESCRIPTIONS: THIS SUBROUTINE WILL SHIFT DE
                  ;                     RIGHT 1 BIT LEAVING THE M.S.B.
                  ;                     EQUAL TO 0.
                  ;*************************************************
0F47  37 SDER:    STC           ;CLEAR
0F48  3F          CMC           ;      CY
0F49  7A          MOV  A,D      ;SHIFT
0F4A  1F          RAR           ;      D
0F4B  57          MOV  D,A      ;      RIGHT
0F4C  7B          MOV  A,E      ;SHIFT
0F4D  1F          RAR           ;      E
0F4E  5F          MOV  E,A      ;      RIGHT
0F4F  C9          RET
```

```
 89                     ;*****************************************************
 90                     ;       (3)     UTILITY SUBROUTINE
 91                     ;       NAME    : RASC
 92                     ;       FUNCTION: I/P FROM TTY AND CONVERT THE
 93                     ;                 NUMBER FROM ASC11 TO HEXDECIMAL.
 94                     ;       INPUTS  : NONE
 95                     ;       OUTPUTS : A, CY
 96                     ;       CALLS   : ECHO, SRIP2
 97                     ;       DESTROYS: STATUS
 98                     ;       DESCRIPTION: THIS ROUTINE SHALL I/P & ECHO
 99                     ;                    A CHARACTER FROM THE TTY TO
100                     ;                    THE ACC.
101                     ;                    IF THE CHARACTER WAS A HEX.
102                     ;                    CHARACTER IT WOULD BE CONVERTED
103                     ;                    TO HEX.
104                     ;                    OTHERWISE, IT SHALL REMAIN IN
105                     ;                    THE FORM OF ASC11 WITH THE CY
106                     ;                    FLAG SET.
107                     ;*****************************************************
108 0F50    CD RASC:    CALL SRIP2      ;I/P CHARACTER FROM TTY
    0F51    5C
    0F52    01
109 0F53    CD          CALL ECHO       ;ECHO TYPE
    0F54    6A
    0F55    0F
110 0F56    FE RASM:    CPI  30
    0F57    30
111 0F58    D8          RC              ;IF < 30H RET
112 0F59    FE          CPI  47
    0F5A    47
113 0F5B    DA          JC   RA1        ;IF < 47H
    0F5C    60
    0F5D    0F
114 0F5E    37          STC             ;SET CY
115 0F5F    C9          RET
116 0F60    FE RA1:     CPI  40
    0F61    40
117 0F62    DA          JC   RA2        ;IF < 40H (I.E. < 10)
    0F63    67
    0F64    0F
118 0F65    C6          ADI  9          ;ADD 9
    0F66    09
119 0F67    E6 RA2:     ANI  0F         ;MASK OUT 1ST 4 BIT
    0F68    0F
120 0F69    C9          RET
121                     ;*****************************************************
122                     ;       (4)     UTILITY SUBROUTINE
123                     ;       NAME    : ECHO
124                     ;       FUNCTION: TO ECHO TYPE A CHARACTER.
125                     ;       INPUTS  : A (CHARACTER IN ASC11)
126                     ;       OUTPUTS : A
127                     ;       CALLS   : SROP2
128                     ;       DESTROYS: ACC. STATUS WORD.
129                     ;       DESCRIPTIONS: THIS ROUTINE WILL TYPE OUT THE
130                     ;                     CHARACTER ON ACC.
131                     ;                     IF IT IS A CR IF WILL ALSO
132                     ;                     TYPE LF AS WELL.
133                     ;*****************************************************
134 0F6A    C5 ECHO:    PUSH B          ;SAVE BC
135 0F6B    4F          MOV  C,A
136 0F6C    CD          CALL SROP2      ;O/P CHARACTER
    0F6D    17
    0F6E    01
137 0F6F    FE          CPI  0D
    0F70    0D
138 0F71    C2          JNZ  EC1        ;IF NOT CR
    0F72    79
    0F73    0F
139 0F74    0E          MVI  C,0A       ;O/P
    0F75    0A
140 0F76    CD          CALL SROP2      ;    LF
    0F77    17
    0F78    01
141 0F79    C1 EC1:     POP  B          ;RESTORE BC
142 0F7A    C9          RET
```

```
                        ;**********************************************
                        ;       (5)     UTILITY SUBROUTINE
                        ;       NAME    : OPSA
                        ;       FUNCTION: TO O/P HL TO TTY AS 4 HEX. DIGITS
                        ;       INPUTS  : HL
                        ;       OUTPUTS : HL
                        ;       CALLS   : CASC
                        ;       DESTROY : A & STATUS
                        ;       DESCRIPTION: THIS ROUTINE OUTPUTS HL TO TTY
                        ;                    IN THE FORM OF FOUR HEX.
                        ;                    DIGITS.
                        ;                    THERE IS ANOTHER POINT OF
                        ;                    ENTRY TO THIS ROUTINE:-
                        ;                    OPSL WHICH O/P L ONLY.
                        ;**********************************************
0F7B    7C      OPSA:   MOV  A,H        ;EXTRACT
0F7C    E6              ANI  0F0        ;       UPPER
0F7D    F0
0F7E    0F              RRC             ;       FOUR
0F7F    0F              RRC             ;       BIT
0F80    0F              RRC             ;       OF
0F81    0F              RRC             ;       H
0F82    CD              CALL CASC       ;O/P 1ST DIGIT
0F83    B1
0F84    0F
0F85    7C              MOV  A,H        ;EXTRACT LOWER
0F86    E6              ANI  0F         ;       4 BIT OF H
0F87    0F
0F88    CD              CALL CASC       ;O/P 2ND DIGIT
0F89    B1
0F8A    0F
0F8B    7D      OPSL:   MOV  A,L        ;EXTRACT
0F8C    E6              ANI  0F0        ;       UPPER
0F8D    F0
0F8E    0F              RRC             ;       FOUR
0F8F    0F              RRC             ;       BIT
0F90    0F              RRC             ;       OF
0F91    0F              RRC             ;       L
0F92    CD              CALL CASC       ;O/P 3RD DIGIT
0F93    B1
0F94    0F
0F95    7D              MOV  A,L        ;EXTRACT LOWER
0F96    E6              ANI  0F         ;       4 BIT OF L
0F97    0F
0F98    CD              CALL CASC       ;O/P 4TH DIGIT
0F99    B1
0F9A    0F
0F9B    C9              RET
                        ;**********************************************
                        ;       (6)     UTILITY SUBROUTINE
                        ;       NAME    : INAT
                        ;       FUNCTION: INPUT A 4 HEX. DIGITS NO. ONTO
                        ;                 DE AND TEST WHETHER THE WHOLE
                        ;                 COMMAND IS ACCEPTED.
                        ;       INPUTS  : NONE
                        ;       OUTPUTS : DE
                        ;       CALLS   : INAD, TYSP, ECHO, SRIP2.
                        ;       DESTROYS: A & STATUS.
                        ;       DESCRIPTION: THIS ROUTINE I/P A 4 DIGIT
                        ;                    HEX. NO. ON TO DE AND TEST
                        ;                    WHETHER THE WHOLE COMMAND
                        ;                    IS ACCEPTED.
                        ;                    THE 4 DIGIT HEX. NO. COULD
                        ;                    BE TYPE IN ANY NO. OF DIGITS.
                        ;**********************************************
0F9C    CD      INAT:   CALL TYSP       ;TYPE ' '
0F9D    DC
0F9E    0F
0F9F    CD              CALL INAD       ;I/P ADDR.
0FA0    03
0FA1    0F
0FA2    FE              CPI  0D
0FA3    0D
0FA4    C8              RZ              ;RETURN IF CR
0FA5    CD      AT1:    CALL SRIP2      ;READ
0FA6    5C
0FA7    01
0FA8    FE              CPI  0D
0FA9    0D
0FAA    C2              JNZ  AT1        ;IF NOT CR
0FAB    A5
0FAC    0F
0FAD    CD              CALL ECHO       ;ECHO TYPE
0FAE    6A
0FAF    0F
0FB0    C9              RET
```

```
                ;*************************************************
                ;       (7)     UTILITY SUBROUTINE
                ;       NAME    : CASC
                ;       FUNCTION: CONVERT A HEX. DIGIT TO ASC11 AND
                ;                 O/P.
                ;       INPUTS  : A
                ;       OUTPUTS : A
                ;       CALLS   : SROP2
                ;       DESTROYS: NOTHING
                ;       DESCRIPTIONS: THIS ROUTINE CONVERTS THE
                ;                     CONTENT OF THE ACC. FORM HEX.
                ;                     TO ASC11 AND O/P THE
                ;                     CHARACTER, THE ACCUMULATOR
                ;                     IS UNCHANGE.
                ;*************************************************
0FB1    F5  CASC:   PUSH PSW        ;SAVE ACC. & STATUS
0FB2    FE          CPI  0A         ;
0FB3    0A
0FB4    DA          JC   CA1        ;IF < 10
0FB5    B9
0FB6    0F
0FB7    C6          ADI  7          ;CONVERT TO
0FB8    07
0FB9    C6  CA1:    ADI  30         ;       ASC11
0FBA    30
0FBB    C5          PUSH B          ;SAVE BC
0FBC    4F          MOV  C,A        ;O/P
0FBD    CD          CALL SROP2      ;    ACC.
0FBE    17
0FBF    01
0FC0    C1          POP  B          ;RESTORE BC
0FC1    F1          POP  PSW        ;RESTORE A
0FC2    C9          RET
```



```
                ;*************************************************
                ;       (8)     UTILITY SUBROUTINE
                ;       NAME    : DRD
                ;       FUNCTION: DISABLE PAPER TAPER READER 2
                ;       INPUTS  : NONE
                ;       OUTPUTS : NONE
                ;       CALLS   : NONE
                ;       DESTROYS: ACC. & STATUS
                ;       DESCRIPTION: THIS PROGRAM CLEARS THE BIT
                ;                    DTR, OF THE COMMAND INSTRUCTION
                ;                    OF TX2 AND ITS BUFFER.
                ;*************************************************
0FC3    3A  DRD:    LDA  0BFC       ;FETCH COMMAND WORD
0FC4    FC
0FC5    0B
0FC6    E6          ANI  0FD        ;CLEAR DTR
0FC7    FD
0FC8    32          STA  0BFC       ;LOAD CONTR. BUFFER
0FC9    FC
0FCA    0B
0FCB    32          STA  8011       ;LOAD COMMAND REGISTER
0FCC    11
0FCD    80
0FCE    C9          RET
                ;*************************************************
                ;       (9)     UTILITY SUBROUTINE
                ;       NAME    : CLX
                ;       FUNCTION: TO CLEAR MEMORIES 09F4 TO 09FD
                ;       INPUTS  : NONE
                ;       OUTPUTS : NONE
                ;       CALLS   : NONE
                ;       DESTROYS: STATUS, A, H, L.
                ;       DESCRIPTION: THIS PROGRAM CLEAR MEMORIES
                ;                    09F4 TO 09FD WHICH ARE THE
                ;                    BUFFER FOR THE PROCESSOR
                ;                    REGISTERS.
                ;*************************************************
0FCF    21  CLX:    LXI  H,09F4     ;LOAD 1ST MEMORY ADDR.
0FD0    F4
0FD1    09
0FD2    AF  CLR:    XRA  A          ;CLEAR ACC.
0FD3    77          MOV  M,A        ;CLEAR MEMORY
0FD4    23          INX  H          ;INC. MEMORY ADDR.
0FD5    7D          MOV  A,L        ;WHETHER REACHING THE
0FD6    EE          XRI  0FE        ;       LAST REG.
0FD7    FE
0FD8    C2          JNZ  CLR
0FD9    D2
0FDA    0F
0FDB    C9          RET
```

```
                        ;*************************************************
                        ;       (10) & (11)  UTILITY SUBROUTINE
                        ;       NAME    : (10) TYSP
                        ;       FUNCTION: (10) TYSP - TYPE SPACE
                        ;       INPUTS  : NONE
                        ;       OUTPUTS : NONE.
                        ;       CALLS   : SROP2
                        ;       DESTROYS: NOTHING.
                        ;       DESCRIPTIONS: AS IN FUNCTION
                        ;*************************************************
0FDC    C5      TYSP:   PUSH B
0FDD    0E              MVI  C,' '
0FDE    20
0FDF    CD              CALL SROP2
0FE0    17
0FE1    01
0FE2    C1              POP  B
0FE3    C9              RET
                        ;*************************************************
                        ;       (12)    UTILITY ROUTINE
                        ;       NAME    : TECH
                        ;       FUNCTION: TYPE 18 'CHARACTER'
                        ;       INPUTS  : C (CHARACTER IN ASC11)
                        ;       OUTPUTS : NONE
                        ;       CALLS   : ECHO
                        ;       DESTROYS: STATUS, A, B, C.
                        ;       DESCRIPTION; AS IN FUNCTION.
                        ;*************************************************
0FE4    06      TECH:   MVI  B,0        ;CLEAR B FOR COUNTER
0FE5    00
0FE6    CD      TE1:    CALL SROP2      ;O/P CHARACTER
0FE7    17
0FE8    01
0FE9    04              INR  B          ;INC. COUNTER
0FEA    78              MOV  A,B        ;FETCH COUNTER
0FEB    FE              CPI  12         ;COMPARE WITH 18
0FEC    12
0FED    DA              JC   TE1        ;COUNTER < 18
0FEE    E6
0FEF    0F
0FF0    C9              RET
                        ;*************************************************
                        ;       (13)    UTILITY SUBROUTINE
                        ;       NAME    : IACS
                        ;       FUNCTION: I/P 2 BYTE ONTO DE AND ADD TO
                        ;                 CHECK SUM INC.
                        ;       INPUTS  : C
                        ;       OUTPUTS : D, E, C.
                        ;       CALLS   : INAD
                        ;       DESTROYS: ACC. & STATUS
                        ;       DESCRIPTION: THIS ROUTINE INPUTS 4 HEX.
                        ;                    DIGIT IN ASC111 FORM AND
                        ;                    CONVERT IT TO 2 BYTE. THEN
                        ;                    THEY ARE STORED IN DE
                        ;                    AND ADD TO CHECK SUM IN C.
                        ;*************************************************
0FF1    CD      IACS:   CALL INAD       ;I/P START ADDR. OF STRING
0FF2    03
0FF3    0F
0FF4    79              MOV  A,C        ;ADD
0FF5    82              ADD  D          ;    TO
0FF6    83              ADD  E          ;    CHECK
0FF7    4F              MOV  C,A        ;    SUM
0FF8    C9              RET
                        ;*************************************************
                        ;       (14)    UTILITY SUBROUTINE
                        ;       NAME    : CR
                        ;       FUNCTION: TYPE 'CR & LF'
                        ;       CALLS   : ECHO
                        ;       DESTROYS: ACC. & STATUS
                        ;*************************************************
0FF9    3E      CR:     MVI  A,0D
0FFA    0D
0FFB    CD              CALL ECHO
0FFC    6A
0FFD    0F
0FFE    C9              RET
                        .END
```

ACKS	0E01	AGAN	04B6	AT1	0FA5	B2	0C31
BACK	07C9	BGLN	0E71	BTWO	0097	C1	0D71
CA1	0FB9	CAID	05ED	CALLI	0727	CASC	0FB1
CCON	0746	CDC	00C6	CER2	018D	CLR	0FD2
CLR1	0256	CLR2	0167	CLX	0FCF	CONV	02BB
COPY	0D54	CPST	0D79	CPTM	0D92	CR	0FF9
CRBK	0D9E	CRLF	040F	CRST	0D8C	CTA	00D3

```
DSCN    0316    DISP    00B9    DLIST   0DC6    DMDA    0E13
DPO     0FC3    DSET    0072    DUMP    0DB4    DVDOP   03B2
E1      0D3B    E2      0D41    E3      0D49    EC1     0F79
ECHO    0F6A    EIP1    0285    EIP2    019D    ENAB1   021F
ENAB2   0138    ENCN    05F7    ENT     0087    ENTER   0D31
ER2     01AB    FER1    0277    FER2    017C    FSTD    0375
GO      0587    HFLI    0CC6    HZOS    0EDB    IACS    0FF1
IND     009F    IN1     0F1A    IN2     0289    IN3     0223
IN4     01C3    IN5     013F    IN6     00DF    IN7     0047
INAD    0F03    INAD1   0F09    INAD2   0F1C    INAD3   0F27
INAT    0F9C    INC     008A    INDT    0D08    INIT    043D
INTRE   0055    IPCT    056F    IPD     0EDF    IPSED   057D
JCON    0702    JMPA    06F0    L1      0E39    L2      0E50
L3      0E5E    L4      0E68    L5      0E6B    L6      0E9C
L7      0E74    L8      0E87    LDD     0EA2    LDT     00AB
LOAD    0E33    LOG     0361    LOOPS1  01EE    LOOPS2  010A
LSP     07E2    LSPR    07E7    M1      0D1A    M2      0D24
MAS1    042A    MAS2    0439    MASK    041A    MODY    0CF5
NEG     0388    NEGE    03AA    NRC     079E    NST     0065
NWR     02A7    NXDA    0D38    NXLI    0CC1    ODR     02AF
ONEB    06C3    OPCT    0565    OPRG    07F1    OPRU    03F8
OPSA    0F7B    OPSL    0F8B    OTX2    0405    OUDA1   01E4
OUDA2   0100    PCHLI   07A6    PEND    00A3    PER2    01A6
POSE    03AC    POSI    038A    PROG    02E5    PZOS    0EB8
RA1     0F60    RA2     0F67    RASC    0F50    RASM    0F56
RCON    077C    RDST    0E47    RDY1    024D    RDY2    015E
RERE    059B    RETA    0770    ROUN    0615    RSTA    07AF
SDER    0F47    SDPY    03D1    SEO     0669    SHDE    0F37
SRIP1   024B    SRIP2   015C    SROP1   01FB    SROP2   0117
STLG    0354    STP     0E70    STRW    0D35    STSF    02E0
STX1    0333    STX2    0331    SXTN    0DFB    T1      05D9
T2      05FA    T4      0642    T5      064D    T6      065B
T7      0699    T8      06AE    T9      0C7F    TBY     0C37
TC      05D6    TC1     0600    TC2     060D    TCR     0551
TE1     0FE6    TECH    0FE4    THRB    06DC    THTER   1000
TICR    059E    TLC     03F0    TNLB    0309    TPC     0C9A
TRAC    05C2    TREN    049D    TST     0057    TSTM    0C70
TSX     0DEF    TUPR    03EA    TWOB    06CC    TYAD    0CF9
TYM     04A7    TYSP    0FDC    USPR    1400    W1      0CDC
WAIT1   01FD    WAIT2   0119    WRI     0CB5    WSAT    0EBC
WSOY    0EB5    X       04FE    X1      04FF    XGO     05AB
YCC     0760    YFR2    0193    YJC     071C    YRC     0793
ZOS     0EDC
*

^C
```

```
                .TITLE "FLOATING POINT ARITHEMATIC"
                .HEX
                .ORG 1000
1000    00      .ORG 1000
1000    C3              JMP  1C00
1001    00
1002    1C
                ;*************************************************
                ;
                ;               ADDITION & SUBTRACTION
                ;       (FLOATING POINT ARITHEMATIC PROGRAM)
                ;
                ;       NAME    : SUMM
                ;       INPUTS  : B, C, D, E, H, L.
                ;       OUTPUTS : B, C, D, E, H, L.
                ;       CALLS   : ROPD, RBCF, CEXP, REXP, SUB, REDL.
                ;
                ;*************************************************
1003    CD      SUMM:   CALL CEXP       ;CONDITION EXP.
1004    3B
1005    13
1006    AF              XRA  A          ;CLEAR ACC.
1007    B0              ORA  B
1008    B1              ORA  C
1009    CA              JZ   RET1       ;IF OPERATOR = 0
100A    7B
100B    10
100C    AF              XRA  A
100D    B2              ORA  D
100E    B3              ORA  E
100F    C2              JNZ  SMI1       ;IF OPERAND NOT= 0
1010    18
1011    10
1012    50              MOV  D,B        ;OPERAND = 0
1013    59              MOV  E,C        ;OPERAND _ OPERATOR
1014    6C              MOV  L,H
1015    C3              JMP  RET1
1016    7D
1017    10
1018    C5      SMI1:   PUSH B          ;SUBROUNINE-FLOATING POINT
1019    7C              MOV  A,H        ;            SUMMATION
101A    E6              ANI  7F
101B    7F
101C    47              MOV  B,A
101D    7D              MOV  A,L
101E    E6              ANI  7F
101F    7F
1020    B8              CMP  B
1021    C1              POP  B
1022    CA              JZ   SM3        ;IF EXP. ARE EQUAL
1023    4C
1024    10
1025    D2              JNC  SM1
1026    2E
1027    10
1028    CD              CALL ROPD       ;IF L< H
1029    C0
102A    12
102B    C3              JMP  SMI1
102C    18
102D    10
102E    79      SM1:    MOV  A,C        ;IF H<L
102F    1F              RAR             ;ROTATE
1030    CD              CALL RBCF       ;       RIGHT 4 TIMES _
1031    02
1032    13
1033    78              MOV  A,B
1034    FE              CPI  50         ;TEST WHETHER ROUND OFF REQ.
1035    50
1036    F5              PUSH PSW        ;     AND SAVE STATUS
1037    E6              ANI  0F         ;CLEAR CARRY OVER DIGIT
1038    0F
1039    47              MOV  B,A        ;       AND RESTORE OPERATOR
103A    F1              POP  PSW        ;BRING BACK STATUS
103B    DA              JC   NCY1       ;       AND BRANCH
103C    48
103D    10
103E    79              MOV  A,C        ;ROUND
103F    C6              ADI  01         ;
1040    01
1041    27              DAA             ;
1042    4F              MOV  C,A        ;       OFF
1043    78              MOV  A,B        ;
1044    CE              ACI  0          ;       AFIER
1045    00
1046    27              DAA             ;
1047    47              MOV  B,A        ;       SHIFT
1048    24      NCY1:   INR  H          ;INC. EXP. OF OPERATOR
1049    C3              JMP  SMI1
104A    18
104B    10
```

```
                3E SM3:   MVI  A,7F    ;TEST EXP., WHETHER IT
                7F
                A5        ANA  L       ;     IS EQUAL
                FE        CPI  7F      ;     TO 127
                7F
                C2        JNZ  SM4     ;EXP. NOT = 127
                5C
                10
                3E        MVI  A,0A0   ;O/P O/F
                A0
                32        STA  8200    ;    ALARM
                00
                82
                C3        JMP  RET1
                7D
                10
                FE SM4:   CPI  0       ;TEST WHETHER EXP. = 0
                00
                C2        JNZ  SM5     ;EXP. NOT = 0
                69
                10
                3E        MVI  A,0A1   ;O/F U/F
                A1
                32        STA  8200    ;    ALARM
                00
                82
                C3        JMP  RET1
                7D
                10
                7D SM5:   MOV  A,L     ;WHETHER SIGN
                AC        XRA  H       ;        ARE EQUAL
                E6        ANI  80      ;        OR DIFFERENT
                80
                C2        JNZ  SUBB    ;SUB IF DIFFERENT
                81
                10
                CD        CALL ADDSR
                82
                12
                D2        JNC  RET1    ;IF NO OVERFLOW
                7D
                10
                CD        CALL ROPD    ;ROTATE RIGHT, ROUND
                C0
                12
                7A        MOV  A,D     ;       OFF & INC. EXP.
                F6        ORI  10      ;       THEN REPLACE
                10
                57        MOV  D,A     ;       CARRY
                CD RET1:  CALL REXP    ;RECONDITION EXP.
                52
                13
                C9        RET
                CD SUBB:  CALL SUBSR
                9B
                12
                D2        JNC  IFZ     ;IF NO CARRY I.E. +VE.
                98
                10
                7D        MOV  A,L     ;COMPLEMENT SIGN
                EE        XRI  80      ;           OF
                80
                6F        MOV  L,A     ;           RESULT
                3E        MVI  A,65    ;CONVERT
                65
                83        ADD  E       ;       RESULT
                2F        CMA            ;
                27        DAA
                5F        MOV  E,A     ;       FROM
                3F        CMC            ;       10'S
                3E        MVI  A,65    ;       COMPLEMENT
                65
                8A        ADC  D       ;       BACK
                2F        CMA            ;       TO
                27        DAA
                57        MOV  D,A     ;       DECIMAL
                3E IFZ:   MVI  A,0     ;TEST
                00
                BB        CMP  E       ;     WHETHER
                C2        JNZ  FSTZ    ;     RESULT
                A2
                10
                BA        CMP  D       ;     IS
                CA        JZ   RET1    ;     ZERO
                7D
                10
                3E FSTZ:  MVI  A,0F0   ;IF 1ST DIGIT
                F0
                A2        ANA  D       ;   IS ZERO
                C2        JNZ  RET1
                7D
                10
```

```
109 10A2  CD          CALL RDEL    ;SHIFT
    10A3  F4
    10A4  12
110 10A5  CD          CALL RDEL    ;      RESULT
    10A6  F4
    10A7  12
111 10A8  CD          CALL RDEL    ;      LEFT
    10A9  F4
    10AA  12
112 10AB  CD          CALL RDEL    ;      ONE DIGIT
    10AC  F4
    10AD  12
113 10AE  2D          DCR  L       ;DEC. EXP.
114 10AF  3E          MVI  A,7F    ;IF U/F
    10B0  7F
115 10B1  A5          ANA  L       ;       OR NOT
116 10B2  C2          JNZ  FSTZ
    10B3  A2
    10B4  10
117 10B5  3E          MVI  A,0A1   ;O/P U/F
    10B6  A1
118 10B7  32          STA  8200    ;    ALARM
    10B8  00
    10B9  82
119 10BA  C9          RET
FLOATING POINT ARITHEMATIC          CALOS-80 V2.12 03/19/71 PAGE    2

  1               ;*********************************************
  2               ;
  3               ;                MULTIPLICATION
  4               ;
  5               ;       (FLOATING POINT ARITHEMATIC PROGRAM)
  6               ;
  7               ;       NAME    : MUPY
  8               ;       INPUTS  : B, C, D, E, H, L.
  9               ;       OUTPUTS : B, C, D, E, H, L.
 10               ;       CALLS   : CEXP, MI, RFOR, REXP, ADD.
 11               ;
 12               ;*********************************************
 13 10BB  3E MUPY:    MVI  A,0     ;CLEAR ACC.
    10BC  00
 14 10BD  B8          CMP  B       ;WHETHER
 15 10BE  C2          JNZ  MU1     ;       OPERATOR
    10BF  C8
    10C0  10
 16 10C1  B9          CMP  C       ;       IS
 17 10C2  C2          JNZ  MU1     ;       ZERO
    10C3  C8
    10C4  10
 18 10C5  50          MOV  D,B     ;SHIFT OPERATOR TO
 19 10C6  59          MOV  E,C     ;      RESULT
 20 10C7  C9          RET
 21 10C8  BA MU1:     CMP  D       ;WHETHER
 22 10C9  C2          JNZ  MU2     ;       OPERAND
    10CA  D1
    10CB  10
 23 10CC  BB          CMP  E       ;       IS
 24 10CD  C2          JNZ  MU2     ;       ZERO
    10CE  D1
    10CF  10
 25 10D0  C9          RET
 26 10D1  CD MU2:     CALL CEXP    ;CONDITION EXP.
    10D2  32
    10D3  13
 27 10D4  E5          PUSH H       ;SAVE BOTH EXPONENT
 28 10D5  7C          MOV  A,H     ;MASK
 29 10D6  E6          ANI  7F      ;
    10D7  7F
 30 10D8  67          MOV  H,A     ;     OUT
 31 10D9  7D          MOV  A,L     ;
 32 10DA  E6          ANI  7F      ;     SIGN BIT
    10DB  7F
 33 10DC  84          ADD  H       ;ADD EXP.
 34 10DD  FE          CPI  0C0     ;IF EXP. >= 128
    10DE  C0
 35 10DF  DA          JC   MU3
    10E0  E9
    10E1  10
 36 10E2  3E          MVI  A,0A2   ;O/P O/F
    10E3  A2
 37 10E4  32          STA  8200    ;    ALARM
    10E5  00
    10E6  82
 38 10E7  E1          POP  H
 39 10E8  C9          RET
 40 10E9  FE MU3:     CPI  40      ;IF EXP. <= 0
    10EA  40
 41 10EB  D2          JNC  MU4
    10EC  F2
    10ED  10
 42 10EE  3E          MVI  A,0A3   ;O/P U/F
    10EF  A3
 43 10F0  32          STA  8200    ;    ALARM
    10F1  00
    10F2  82
```

```
44 10FF    E1          POP  H
45 1100    C9          RET
46 1101    EE 80 MU4:  XRI  80       ;COMPL. SIGN OF EXP.
   1102    80
47 1103    E6          ANI  7F       ;CLEAR 1ST BIT
   1104    7F
48 1105    6F          MOV  L,A      ;STORE EXP. OF RESULT IN L
49 1106    E3          XTHL          ;RESTORE SIGNS
50 1107    7C          MOV  A,H      ;EXTRACT
51 1108    F6          ORI  7F       ;        SIGNS
   1109    7F
52 110A    67          MOV  H,A
53 110B    7B          MOV  A,L
54 110C    E6          ANI  80
   110D    80
55 110E    AC          XRA  H        ;WHETHER SIGNS ARE EQUAL
56 110F    2F          CMA           ;SET SIGN '1'(+VE) IF EQUAL
57 1110    E1          POP  H        ;RESTORE EXP. OF RESULT
58 1111    B5          ORA  L        ;STORE
59 1112    6F          MOV  L,A      ;      SIGN OF RESULT
60 1113    E5          PUSH H        ;EXP. ONTO STACK
61 1114    60          MOV  H,B      ;MUTIPLIER
62 1115    69          MOV  L,C      ;          TO HL
63 1116    42          MOV  B,D      ;MUTIPLICAND
64 1117    4B          MOV  C,E      ;            TO BC
65 1118    16          MVI  D,0      ;CLEAR
   1119    00
66 111A    1E          MVI  E,0      ;      DE
   111B    00
67 111C    7C          MOV  A,H      ;EXTRACT
68 111D    E6          ANI  0F0      ;        1ST DIGIT
   111E    F0
69 111F    0F          RRC           ;
70 1120    0F          RRC           ;        OF
71 1121    0F          RRC           ;
72 1122    0F          RRC           ;        MULTIPLIER
73 1123    E5          PUSH H        ;MULTIPLIER TO STACK
74 1124    67          MOV  H,A      ;1ST DIGIT OF MULTIPLIER IN H
75 1125    2E          MVI  L,0      ;CLEAR L
   1126    00
76 1127    CD          CALL MI       ;MUPY 1ST DIGIT OF MULTIPLIER
   1128    0F
   1129    13
77                                   ;     WITH MUPLICAND
78 112A    E3          XTHL          ;O/F TO STACK MUPLIER TO HL
79 112B    D5          PUSH D        ;1ST PRODUCT TO STACK
80 112C    16          MVI  D,0      ;CLEAR
   112D    00
81 112E    1E          MVI  E,0      ;      DE
   112F    00
82 1130    7C          MOV  A,H      ;EXTRACT 2ND DIGIT
83 1131    E6          ANI  0F       ;        OF MULTIPLIER
   1132    0F
84 1133    E5          PUSH H        ;STORE MULTIPLIER IN STACK
85 1134    67          MOV  H,A      ;PUT 2ND DIGIT IN H
86 1135    2E          MVI  L,0      ;CLEAR L
   1136    00
87 1137    CD          CALL MI       ;MUPY 2ND DIGIT WITH
   1138    0F
   1139    13
88                                   ;     MUPLICAND
89 113A    E3          XTHL          ;O/F TO STACK, MUPLIER TO HL
90 113B    D5          PUSH D        ;2ND PRODUT TO STACK
91 113C    16          MVI  D,0      ;CLEAR
   113D    00
92 113E    1E          MVI  E,0      ;      DE
   113F    00
93 1140    7D          MOV  A,L      ;EXTRACT
94 1141    E6          ANI  0F0      ;        3RD DIGIT
   1142    F0
95 1143    0F          RRC           ;
96 1144    0F          RRC           ;        OF
97 1145    0F          RRC           ;
98 1146    0F          RRC           ;        MULTIPLIER
99 1147    E5          PUSH H        ;MULTIPLIER TO STACK
100 1148   67          MOV  H,A      ;3RD DIGIT IN H
101 1149   2E          MVI  L,0      ;CLEAR L FOR O/F
   114A    00
102 114B   CD          CALL MI       ;MUPY 3RD DIGIT OF MUPLIER
   114C    0F
   114D    13
103                                  ;     WITH MUPLICAND
104 114E   E3          XTHL          ;O/F TO STACK, MUPLIER TO HL
105 114F   D5          PUSH D        ;3RD PRODUCT TO STACK
106 1150   16          MVI  D,0      ;CLEAR
   1151    00
107 1152   1E          MVI  E,0      ;      DE
   1153    00
108 1154   7D          MOV  A,L      ;EXTRACT 4TH DIGIT
109 1155   E6          ANI  0F       ;        OF MULTIPLIER
   1156    0F
110 1157   E5          PUSH H        ;STORE MULTIPLIER IN STACK
111 1158   67          MOV  H,A      ;PUT 4TH DIGIT IN H
```

```
1152 1153    2E          MVI  L,0      ;CLEAR L FOR O/F
     1154    00
1153 1155    CD          CALL MI       ;MUPY 4TH DIGIT OF MULTIPLIER
     1156    0F
     1157    13
1154                                   ;     WITH MULTIPLICAND
1155            ;ADD 4TH PRODUCT WITH 3RD
1156 1158    CD          CALL RFOR     ;SHIFT PREVIOUS RESULT RIGHT
     1159    2D
     115A    13
1157 115B    E1          POP  H        ;FETCH 2ND RESULT FROM
1158 115C    C1          POP  B        ;      STACK ONTO BC
1159 115D    E3          XTHL          ;      AND O/F ONTO L
1160 115E    CD          CALL ADDSR    ;ADD BC&DE TOGETHER
     115F    82
     1160    12
1161 1161    D2          JNC  MU5
     1162    65
     1163    11
1162 1164    2C          INR  L        ;INC. PRODUCT IF CY
1163            ;ADD SUM WITH 2ND PRODUCT
1164 1165    CD MU5:     CALL RFOR
     1166    2D
     1167    13
1165 1168    E1          POP  H
1166 1169    C1          POP  B
1167 116A    E3          XTHL
1168 116B    CD          CALL ADDSR
     116C    82
     116D    12
1169 116E    D2          JNC  MU6
     116F    72
     1170    11
1170 1171    2C          INR  L
1171            ;ADD SUM WITH 1ST PRODUCT
1172 1172    CD MU6:     CALL RFOR
     1173    2D
     1174    13
1173 1175    E1          POP  H
1174 1176    C1          POP  B
1175 1177    E3          XTHL
1176 1178    CD          CALL ADDSR
     1179    82
     117A    12
1177 117B    D2          JNC  MU7
     117C    7F
     117D    11
1178 117E    2C          INR  L
1179 117F    C1 MU7:     POP  B        ;RESTORE MUPLIER ON BC
1180 1180    7D          MOV  A,L      ;WHETHER O/F DIGIT
1181 1181    FE          CPI  0        ;       IS ZERO
     1182    00
1182 1183    C2          JNZ  MU8      ;NO
     1184    94
     1185    11
1183 1186    E1          POP  H        ;RESTORE EXP.
1184 1187    2D          DCR  L        ;DECREMENT EXP. OF RESULT
1185 1188    7D          MOV  A,L      ;WHETHER
1186 1189    E6          ANI  7F       ;     EXP. IS ZERO
     118A    7F
1187 118B    C2          JNZ  TRM
     118C    AA
     118D    11
1188 118E    3E          MVI  A,0A3    ;O/P
     118F    A3
1189 1190    32          STA  8200     ;    U/F ALARM
     1191    00
     1192    82
1190 1193    C9          RET
1191 1194    CD MU8:     CALL RFOR     ;SHIFT RESULT RIGHT
     1195    2D
     1196    13
1192 1197    1F          RAR           ;      ONE DIGIT
1193 1198    FE          CPI  50       ;WHETHER A>=50
     1199    50
1194 119A    D2          JNC  MU9      ;YES
     119B    A0
     119C    11
1195 119D    C3          JMP  TRM      ;JUMP TO END
     119E    A9
     119F    11
1196 11A0    7B MU9:     MOV  A,E      ;ADD
1197 11A1    3C          INR  A        ;    ONE
1198 11A2    27          DAA           ;    TO
1199 11A3    5F          MOV  E,A      ;    RESULT
1200 11A4    7A          MOV  A,D      ;    DUE
1201 11A5    CE          ACI  0        ;    TO
     11A6    00
1202 11A7    27          DAA           ;    ROUND
1203 11A8    57          MOV  D,A      ;    OFF
1204 11A9    E1 TRM:     POP  H        ;RESTORE EXP.
1205 11AA    CD TRM:     CALL REXP     ;RECONDITION EXP. OF RESULT
     11AB    52
     11AC    13
1206 11AD    C9          RET
```

```
                ;***********************************************
                ;
                ;               DIVISION
                ;
                ;       (FLOATING POINT ARITHEMATIC PROGRAM)
                ;
                ;       NAME    : DVD
                ;       INPUTS  : B, C, D, E, H, L.
                ;       OUTPUTS : B, C, D, E, H, L.
                ;       CALLS   : CEXP, SUB, ADD, RBCF, RDEL, RHLL,
                ;                 REXP.
                ;
                ;***********************************************
11AE  AF  DVD:  XRA  A          ;DIVISION SUBROUTINE
11AF  B8        CMP  B          ;WHETHER OPERATOR
11B0  C2        JNZ  DV1        ;       IS
11B1  BD
11B2  11
11B3  B9        CMP  C          ;       EQUAL
11B4  C2        JNZ  DV1        ;       ZERO
11B5  BD
11B6  11
11B7  3E        MVI  A,0A6      ;O/P INF
11B8  A6
11B9  32        STA  8200       ;    AND ALARM
11BA  00
11BB  82
11BC  C9        RET
11BD  BA  DV1:  CMP  D          ;WHETHER
11BE  C2        JNZ  DV2        ;       OPERAND
11BF  C3
11C0  11
11C1  BB        CMP  E          ;       IS
11C2  C8        RZ              ;       ZERO
11C3  E5  DV2:  PUSH H          ;COPY EXP. ONTO STACK
11C4  7C        MOV  A,H        ;EXTRACT
11C5  F6        ORI  7F         ;       SIGN
11C6  7F
11C7  67        MOV  H,A
11C8  7D        MOV  A,L
11C9  E6        ANI  80
11CA  80
11CB  AC        XRA  H          ;COMPARE BOTH SIGNS
11CC  2F        CMA             ;SET SIGN '1' (+VE) IF EQUAL
11CD  6F        MOV  L,A        ;STORE SIGN OF RESULT
11CE  E3        XTHL            ;STORE SIGN IN STACK AND
                                ;       RESTORE EXP.
11CF  CD        CALL CEXP       ;CONDN. EXP.
11D0  3B
11D1  13
11D2  7C        MOV  A,H        ;EXTRACT EXP. OF
11D3  E6        ANI  7F         ;       DIVIDOR
11D4  7F
11D5  67        MOV  H,A
11D6  7D        MOV  A,L        ;EXTRACT EXP. OF
11D7  E6        ANI  7F         ;       DIVIDANT
11D8  7F
11D9  94        SUB  H          ;SUBTRACT EXP. OF DIVIDANT
                                ;       BY DIVIDOR
11DA  FA        JM   DV3
11DB  EB
11DC  11
11DD  C6        ADI  40
11DE  40
11DF  FE        CPI  80         ;IF EXP. >= 128
11E0  80
11E1  DA        JC   DV4        ;NO
11E2  F9
11E3  11
11E4  3E        MVI  A,0A4      ;O/P O/F
11E5  A4
11E6  32        STA  8200       ;    ALARM
11E7  00
11E8  82
11E9  E1        POP  H
11EA  C9        RET
11EB  C6  DV3:  ADI  40
11EC  40
11ED  FE        CPI  80         ;IF EXP.<=0
11EE  80
11EF  DA        JC   DV4        ;NO
11F0  F9
11F1  11
11F2  3E        MVI  A,0A5      ;O/P U/F
11F3  A5
11F4  32        STA  8200       ;    ALARM
11F5  00
11F6  82
11F7  E1        POP  H
11F8  C9        RET
```

```
60 11F9  6F DV4:   MOV  L,A     ;STORE EXP. OF RESULT IN L
61 11FA  E3        XTHL         ;EXP. TO STACK & SIGN TO HL
62 11FB  7D        MOV  A,L     ;EXTRACT
63 11FC  E6        ANI  80      ;       SIGN OF RESULT
   11FD  80
64 11FE  E1        POP  H       ;RESTORE EXP. OF RESULT
65 11FF  B5        ORA  L       ;ATTACH
66 1200  6F        MOV  L,A     ;       SIGN OF RESULT
67 1201  E5        PUSH H       ;STORE EXP. IN STACK
68 1202  26        MVI  H,0     ;CLEAR
   1203  00
69 1204  2E        MVI  L,0     ;       H & L
   1205  00
70 1206  CD RESU:  CALL SUBSR   ;DIVIDANT SUBTRACTED BY DIVIDER
   1207  9B
   1208  12
71 1209  DA        JC   DV5     ;IF RESULT -VE
   120A  10
   120B  12
72 120C  2C        INR  L       ;INCREMENT QUOTIENT
73 120D  C3        JMP  RESU    ;REPEAT SUBTRACTION
   120E  06
   120F  12
74 1210  CD DV5:   CALL ADDSR   ;RESTORE THE DIVIDANT TO +VE
   1211  B2
   1212  12
75 1213  3E        MVI  A,0     ;WHETHER
   1214  00
76 1215  BC        CMP  H       ;       QUOTIENT
77 1216  C2        JNZ  DEND    ;       IS
   1217  2A
   1218  12
78 1219  BD        CMP  L       ;       ZERO
79 121A  C2        JNZ  DEND
   121B  2A
   121C  12
80 121D  E3        XTHL         ;DECREMENT
81 121E  2D        DCR  L       ;         EXP. OF
82 121F  E3        XTHL         ;         RESULT
83 1220  CD        CALL RBCF    ;SHIFT DIVIDER RIGHT ONE DIGIT
   1221  02
   1222  13
84 1223  78        MOV  A,B     ;CLEAR
85 1224  E6        ANI  0F      ;       CARRY OVER
   1225  0F
86 1226  47        MOV  B,A     ;       DIGIT FROM ROTATION
87 1227  C3        JMP  RESU    ;DIVIDING ANOTHER DIGIT
   1228  06
   1229  12
88 122A  78 DEND:  MOV  A,B     ;IF FIRST DIGIT OF DIVIDER
89 122B  E6        ANI  0F0     ;   IS ZERO
   122C  F0
90 122D  CA        JZ   DV6     ;YES
   122E  3A
   122F  12
91 1230  CD        CALL RBCF    ;SHIFT BC RIGHT ONE DIGIT
   1231  02
   1232  13
92 1233  78        MOV  A,B     ;CLEAR
93 1234  E6        ANI  0F      ;       CARRY OVER
   1235  0F
94 1236  47        MOV  B,A     ;       DIGIT FROM ROTATION
95 1237  C3        JMP  DV7
   1238  4A
   1239  12
96 123A  CD DV6:   CALL RDEL    ;SHIFT
   123B  F4
   123C  12
97 123D  CD        CALL RDEL    ;       DE
   123E  F4
   123F  12
98 1240  CD        CALL RDEL    ;       LEFT
   1241  F4
   1242  12
99 1243  CD        CALL RDEL    ;       ONE DIGIT
   1244  F4
   1245  12
100 1246 7B        MOV  A,E     ;CLEAR CARRY OVER
101 1247 E6        ANI  0F0     ;       DIGIT FROM
   1248  F0
102 1249 5F        MOV  E,A     ;       ROTATION
103 124A 7C DV7:   MOV  A,H     ;IF 1ST. DIGIT OF QUOTIENT
104 124B E6        ANI  0F0     ;   IS STILL ZERO
   124C  F0
105 124D C2        JNZ  DV8     ;NO
   124E  63
   124F  12
106 1250 CD        CALL RHLL    ;SHIFT
   1251  FB
   1252  12
107 1253 CD        CALL RHLL    ;       QUOTIENT
   1254  FB
   1255  12
```

```
108 1256  CD        CALL RHLL    ;      LEFT
    1257  FB
    1258  12
109 1259  CD        CALL RHLL    ;      4
    125A  FB
    125B  12
110 125C  7D        MOV  A,L     ;CLEAR CARRY OVER
111 125D  E6        ANI  0F0     ;      DIGIT FROM
    125E  F0
112 125F  6F        MOV  L,A     ;      ROTATION
113 1260  C3        JMP  RESU    ;DIVIDING ANOTHER DIGIT
    1261  06
    1262  12
114 1263  EB DV8:    XCHG         ;PUT QUOTIENT ONTO DE
115 1264  E1        POP  H       ;RESTORE EXP.
116 1265  2C        INR  L       ;INCREMENT EXP.
117 1266  7D        MOV  A,L
118 1267  E6        ANI  7F      ;IF U/F
    1268  7F
119 1269  C2        JNZ  DV9     ;NO
    126A  72
    126B  12
120 126C  3E        MVI  A,0A5   ;O/P U/F
    126D  A5
121 126E  32        STA  8200    ;    ALARM
    126F  00
    1270  82
122 1271  C9        RET
123 1272  2F DV9:   CMA          ;IF
124 1273  E6        ANI  7F      ;   O/F
    1274  7F
125 1275  C2        JNZ  DV10    ;NO
    1276  7E
    1277  12
126 1278  3E        MVI  A,0A4   ;O/P O/F
    1279  A4
127 127A  32        STA  8200    ;    ALARM
    127B  00
    127C  82
128 127D  C9        RET
129 127E  CD DV10:  CALL REXP    ;RECONDITION EXP. OF RESULT
    127F  52
    1280  13
130 1281  C9        RET          ;FINISH
```



```
  1           ;*************************************************
  2           ;       (1)     UTILITY SUBROUTINE (FOR SUMM)
  3           ;
  4           ;       NAME    : ADDSR
  5           ;       FUNCTION: B.C.D. ADDITION OF BC AND
  6           ;                 DE WITH RESULT IN DE
  7           ;       INPUTS  : B, C, D, E.
  8           ;       OUTPUTS : D, E, STATUS.
  9           ;       DESTROYS: ACC.
 10           ;       CALLS   : NONE
 11           ;*************************************************
 12 1282  7B ADDSR:  MOV  A,E
 13 1283  81        ADD  C
 14 1284  DA        JC   AD1     ;RETAIN CY IF SET
    1285  8B
    1286  12
 15 1287  27        DAA          ;B.C.D. CORRECTION
 16 1288  C3        JMP  AD2
    1289  8D
    128A  12
 17 128B  27 AD1:   DAA
 18 128C  37        STC
 19 128D  5F AD2:   MOV  E,A     ;STORE RESULT IN E
 20 128E  7A        MOV  A,D     ;FETCH 3RD. & 4TH. DIGIT
 21 128F  88        ADC  B
 22 1290  DA        JC   AD3     ;RETAIN CY IF SET
    1291  97
    1292  12
 23 1293  27        DAA
 24 1294  C3        JMP  AD4
    1295  99
    1296  12
 25 1297  27 AD3:   DAA
 26 1298  37        STC
 27 1299  57 AD4:   MOV  D,A     ;STORE RESULT IN D
 28 129A  C9        RET
 29           ;*************************************************
 30           ;       (2)     UTILITY SUBROUTINE (FOR SUMM)
 31           ;
 32           ;       NAME    : SUBSR
 33           ;       FUNCTION: B.C.D. SUBTRACTION OF BC AND
 34           ;                 DE (DE-BC) WITH RESULT IN DE
 35           ;       INPUTS  : B, C, D, E.
 36           ;       OUTPUTS : D, E & STATUS
 37           ;       DESTROYS: ACC.
 38           ;       CALLS   : NONE
 39           ;*************************************************
```

```
 40 129B    3E SUBSR:  MVI  A,65     ;10'S
    129C    65
 41 129D    81         ADD  C        ;     COMPLEMENT
 42 129E    2F         CMA           ;     C
 43 129F    83         ADD  E        ;E SUBTRACTED BY C
 44 12A0    F5         PUSH PSW      ;SAVE CY
 45 12A1    27         DAA
 46 12A2    5F         MOV  E,A      ;STORE RESULT IN E
 47 12A3    DA         JC   SB1      ;RESTORE
    12A4    AA
    12A5    12
 48 12A6    F1         POP  PSW      ;        PREVIOUS
 49 12A7    C3         JMP  SB2      ;        CY
    12A8    AC
    12A9    12
 50 12AA    F1 SB1:    POP  PSW      ;        FLAG
 51 12AB    37         STC           ;        FOR
 52 12AC    3F SB2:    CMC           ;        2ND BYTE
 53 12AD    3E         MVI  A,65     ;10'S
    12AE    65
 54 12AF    88         ADC  B        ;     COMPLEMENT
 55 12B0    2F         CMA           ;     B + CY
 56 12B1    82         ADD  D        ;D-B
 57 12B2    F5         PUSH PSW      ;SAVE CY
 58 12B3    27         DAA
 59 12B4    57         MOV  D,A      ;STORE RESULT IN D
 60 12B5    DA         JC   SB3      ;RESTORE
    12B6    BC
    12B7    12
 61 12B8    F1         POP  PSW      ;        CY
 62 12B9    C3         JMP  SB4      ;        FLAG
    12BA    BE
    12BB    12
 63 12BC    F1 SB3:    POP  PSW      ;        FOR
 64 12BD    37         STC           ;        FUTHER
 65 12BE    3F SB4:    CMC           ;        USE
 66 12BF    C9         RET
 67                    ;*********************************************
 68                    ;       (3)     UTILITY SUBROUTINE (FOR SUMM)
 69                    ;
 70                    ;       NAME     : ROPD
 71                    ;       FUNCTION: TO ROUND OFF A NUMBER STORED
 72                    ;                 IN DE IF CY HAS BEEN SET
 73                    ;       INPUTS   : D, E, STATUS WORD
 74                    ;       OUTPUTS  : D, E.
 75                    ;       DESTROYS: ACC. & STATUS WORD
 76                    ;       CALLS    : RDER
 77                    ;*********************************************
 78 12C0    7B ROPD:   MOV  A,E      ;ROTATE
 79 12C1    1F         RAR           ;
 80 12C2    CD         CALL RDER     ;
    12C3    ED
    12C4    12
 81 12C5    CD         CALL RDER     ;       RIGHT
    12C6    ED
    12C7    12
 82 12C8    CD         CALL RDER     ;
    12C9    ED
    12CA    12
 83 12CB    CD         CALL RDER     ;       4 TIMES
    12CC    ED
    12CD    12
 84 12CE    7A         MOV  A,D      ;FETCH CARRY OVER & FIRST DIGIT
 85 12CF    F5         PUSH PSW      ;SAVE CARRY OVER DIGIT
 86 12D0    E6         ANI  0F       ;CLEAR CARRY OVER DIGIT
    12D1    0F
 87 12D2    57         MOV  D,A      ;       AND RESTORE OPERAND
 88 12D3    F1         POP  PSW
 89 12D4    1F         RAR
 90 12D5    FE         CPI  50
    12D6    50
 91 12D7    DA         JC   NCY2     ;IF ROUND OFF NOT REQUIRE
    12D8    E4
    12D9    12
 92 12DA    7B         MOV  A,E      ;ROUND
 93 12DB    C6         ADI  01       ;
    12DC    01
 94 12DD    27         DAA           ;       OFF
 95 12DE    5F         MOV  E,A      ;
 96 12DF    7A         MOV  A,D      ;       AFTER
 97 12E0    CE         ACI  0        ;
    12E1    00
 98 12E2    27         DAA           ;
 99 12E3    57         MOV  D,A      ;       SHIFT
100 12E4    2C NCY2:   INR  L        ;INC. EXP. OF OPERAND
101 12E5    C9         RET
```

```
                ;*************************************************
                ;       (4)     UTILITY SUBROUTINES (ROTATION)
                ;
                ;       NAME    : (A) RBCR
                ;                 (B) RDER
                ;                 (C) RDEL
                ;                 (D) RHLL
                ;                 (E) RBCF
                ;       FUNCTION: (A) ROTATE BC RIGHT 1 BIT
                ;                 (B) ROTATE DE RIGHT 1 BIT
                ;                 (C) ROTATE DE LEFT 1 BIT
                ;                 (D) ROTATE HL LEFT 1 BIT
                ;                 (E) ROTATE BC RIGHT 4 BIT
                ;       INPUTS  : (A) BC
                ;                 (B) & (C) DE
                ;                 (D) HL
                ;                 (E) BC
                ;       OUTPUTS : (A) BC
                ;                 (B) & (C) DE
                ;                 (D) HL
                ;                 (E) BC
                ;       DISTROYS: ACC. & STATUS
                ;       CALLS   : (A) - (D) NONE
                ;                 (E) RBCR
                ;*************************************************
12E6  78 RBCR:  MOV  A,B
12E7  1F        RAR
12E8  47        MOV  B,A
12E9  79        MOV  A,C
12EA  1F        RAR
12EB  4F        MOV  C,A
12EC  C9        RET
12ED  7A RDER:  MOV  A,D
12EE  1F        RAR
12EF  57        MOV  D,A
12F0  7B        MOV  A,E
12F1  1F        RAR
12F2  5F        MOV  E,A
12F3  C9        RET
12F4  7B RDEL:  MOV  A,E
12F5  17        RAL
12F6  5F        MOV  E,A
12F7  7A        MOV  A,D
12F8  17        RAL
12F9  57        MOV  D,A
12FA  C9        RET
12FB  7D RHLL:  MOV  A,L
12FC  17        RAL
12FD  6F        MOV  L,A
12FE  7C        MOV  A,H
12FF  17        RAL
1300  67        MOV  H,A
1301  C9        RET
1302  CD RBCF:  CALL RBCR
1303  E6
1304  12
1305  CD        CALL RBCR
1306  E6
1307  12
1308  CD        CALL RBCR
1309  E6
130A  12
130B  CD        CALL RBCR
130C  E6
130D  12
130E  C9        RET
```



```
                ;*************************************************
                ;       (5)     UTILITY SUBROUTINE (FOR MUPY)
                ;
                ;       NAME    : MI
                ;       FUNCTION: MULTIPLY BC BY H WITH OVER FLOW
                ;                 STORE IN L
                ;       INPUTS  : B, C, D, E, H
                ;       OUTPUTS : D, E, L
                ;       DESTROYS: H, ACC. & STATUS WORD
                ;       CALLS   : ADD
                ;*************************************************
130F  2E MI:    MVI  L,0
1310  00
1311  3E MT:    MVI  A,0
1312  00
1313  BC        CMP  H
1314  C2        JNZ  MT1      ;IF H NOT = 0
1315  18
1316  13
1317  C9        RET
1318  CD MT1:   CALL ADDSR
1319  82
131A  12
```

```
18 131B   3E          MVI  A,0      ;ADD CY
   131C   00
19 131D   8D          ADC  L        ;    TO
20 131E   6F          MOV  L,A      ;    L
21 131F   25          DCR  H        ;DEC. MULTIPLIER
22 1320   C3          JMP  MT
   1321   11
   1322   13
23                    ;*************************************************
24                    ;       (6)     UTILITY SUBROUTINE (FOR MUPY)
25                    ;
26                    ;       NAME    : RI
27                    ;       FUNCTION: TO SHIFT OVERFLOW STORED IN
28                    ;                 ACC. INTO DE
29                    ;       INPUTS  : ACC., D, E, L
30                    ;       OUTPUTS : D, E, L
31                    ;       DESTROYS: ACC., STATUS WORD, L
32                    ;       CALLS   : NONE
33                    ;*************************************************
34 1323   1F RI:      RAR
35 1324   6F          MOV  L,A
36 1325   7A          MOV  A,D
37 1326   1F          RAR
38 1327   57          MOV  D,A
39 1328   7B          MOV  A,E
40 1329   1F          RAR
41 132A   5F          MOV  E,A
42 132B   7D          MOV  A,L
43 132C   C9          RET
44                    ;*************************************************
45                    ;       (7)     UTILITY SUBROUTINE (FOR MUPY)
46                    ;
47                    ;       NAME    : RFOR
48                    ;       FUNCTION: SHIFT 4 TIMES OVERFLOW TO DE
49                    ;       CALLS   : RI
50                    ;*************************************************
51 132D   7D RFOR:    MOV  A,L
52 132E   CD          CALL RI
   132F   23
   1330   13
53 1331   CD          CALL RI
   1332   23
   1333   13
54 1334   CD          CALL RI
   1335   23
   1336   13
55 1337   CD          CALL RI
   1338   23
   1339   13
56 133A   C9          RET
57                    ;*************************************************
58                    ;       (8)     UTILITY SUBROUTINE
59                    ;
60                    ;       NAME    : CEXP
61                    ;       FUNCTION: TO CHANGE THE EXPONENT FROM A
62                    ;                 SIGNED BINARY NUMBER TO A
63                    ;                 7 BIT PURE BINARY
64                    ;       INPUTS  : H, L
65                    ;       OUTPUTS : H, L
66                    ;       DESTROYS: ACC. & STATUS WORD
67                    ;       CALLS   : NONE
68                    ;*************************************************
69 133B   7C CEXP:    MOV  A,H      ;CONDN. OPERATOR
70 133C   E6          ANI  40
   133D   40
71 133E   C2          JNZ  NAC1     ;IF SIGN OF EXP. = '1'
   133F   46
   1340   13
72 1341   7C          MOV  A,H      ;COMPLEMENT EXP. BUT
73 1342   EE          XRI  3F       ;           NOT SIGN
   1343   3F
74 1344   3C          INR  A        ;CHANGE TO 2'S COMPL.
75 1345   67          MOV  H,A      ;RESTORE H
76 1346   7D NAC1:    MOV  A,L      ;CONDN. OPERAND
77 1347   E6          ANI  40
   1348   40
78 1349   C2          JNZ  NAC2
   134A   51
   134B   13
79 134C   7D          MOV  A,L
80 134D   EE          XRI  3F
   134E   3F
81 134F   3C          INR  A
82 1350   6F          MOV  L,A
83 1351   C9 NAC2:    RET
```

```
                ;*************************************************
                ;       (9)     UTILITY SUBROUTINE
                ;
                ;       NAME    : REXP
                ;       FUNCTION: TO RESTORE THE EXPONENT OF THE
                ;                 RESULT TO A SIGNED BINARY NO.
                ;       INPUTS  : L
                ;       OUTPUTS : L
                ;       DESTROYS: ACC. & STATUS WORD
                ;       CALLS   : NONE
                ;*************************************************
1352  7D  REXP:   MOV  A,L
1353  E6          ANI  40
1354  40
1355  C2          JNZ  NAC3     ;IF M.S.B. OF EXP. = '1'
1356  5D
1357  13
1358  7D          MOV  A,L
1359  EE          XRI  3F       ;2'S COMPLEMENT
135A  3F
135B  3C          INR  A        ;    EXCEPT SIGNS
135C  6F          MOV  L,A      ;RESTORE L
135D  C9  NAC3:   RET
FLOATING POINT ARITHEMATIC          CALOS-80 V2.12 03/19/71 PAGE    6

                ;
                ;NEW SECTION: "BINARY ARITHEMATIC PROGRAMS"
                ;
                ;*****************************************
                ;
                ;       THESE ROUTINES PERFORM INTEGER BINARY
                ;   ARITHEMATICS.
                ;
                ;*****************************************
                ;=================================
                ;       NAME:   BSU
                ;       INPUT:  HL(SUBTRACTANT), DE(SUBTRACTOR).
                ;       OUTPUT: HL, CY.
                ;       DESTROYS:ACC. & STATUS.
                ;       DESCRIPTION:I.E.  HL-DE=HL
                ;                   IF CARRY IS SET, THE RESULT IS -VE.
                ;=================================================
135E  7D  BSU:    MOV     A,L
135F  93          SUB     E
1360  6F          MOV     L,A
1361  7C          MOV     A,H
1362  9A          SBB     D
1363  67          MOV     H,A
1364  C9          RET
                ;=================================================
                ;       NAME:   BMU
                ;       INPUT:  HL(MULTIPLICANT), DE(MULTIPLIER).
                ;       OUTPUT: HL, CY.
                ;       CALLS:  OVFM.
                ;       DESTROYS:ACC. & STATUS.
                ;       DESCRIPTION:HL * DE = HL
                ;=================================================
1365  C5  BMU:    PUSH    B
1366  D5          PUSH    D
1367  44          MOV     B,H
1368  4D          MOV     C,L
1369  AF          XRA     A
136A  67          MOV     H,A
136B  6F          MOV     L,A
136C  E5          PUSH    H
136D  26          MVI     H,0F
136E  0F
136F  7A  SBM:    MOV     A,D
1370  1F          RAR
1371  57          MOV     D,A
1372  7B          MOV     A,E
1373  1F          RAR
1374  5F          MOV     E,A
1375  D2          JNC     NOK
1376  7E
1377  13
1378  E3          XTHL
1379  09          DAD     B
137A  DC          CC      OVFM
137B  99
137C  13
137D  E3          XTHL
137E  AF  NOK:    XRA     A       ;CLEAR CY FLAG
137F  79          MOV     A,C
1380  17          RAL
1381  4F          MOV     C,A
1382  78          MOV     A,B
1383  17          RAL
1384  47          MOV     B,A
1385  D2          JNC     MZ
1386  91
1387  13
```

```
 61 1388    AF          XRA     A
 62 1389    B2          ORA     D
 63 138A    C4          CNZ     OVFM
    138B    99
    138C    13
 64 138D    B3          ORA     E
 65 138E    C4          CNZ     OVFM
    138F    99
    1390    13
 66 1391    25 MZ:      DCR     H
 67 1392    C2          JNZ     SBM
    1393    6F
    1394    13
 68 1395    E1          POP     H
 69 1396    D1          POP     D
 70 1397    C1          POP     B
 71 1398    C9          RET
 72                ;SUBROUTINE FOR OVERFLOW ALARM.
 73 1399    3E OVFM:    MVI     A,0A9
    139A    A9
 74 139B    32          STA     8200
    139C    00
    139D    82
 75 139E    C9          RET
 76                ;=======================================
 77                ;       NAME:   BDV
 78                ;       INPUT:  HL(DIVIDANT), DE(DIVIDER).
 79                ;       OUTPUT: HL
 80                ;       CALLS:  OVFM.
 81                ;       DESTROYS:ACC. & STATUS.
 82                ;       DESCRIPTION:HL / DE = HL.
 83                ;=======================================
 84 139F    C5 BDV:     PUSH    B
 85 13A0    01          LXI     B,0
    13A1    00
    13A2    00
 86 13A3    AF          XRA     A
 87 13A4    BA          CMP     D
 88 13A5    C2          JNZ     BDV1
    13A6    B1
    13A7    13
 89 13A8    BB          CMP     E
 90 13A9    C2          JNZ     BDV1
    13AA    B1
    13AB    13
 91 13AC    CD          CALL    OVFM
    13AD    99
    13AE    13
 92 13AF    C1          POP     B
 93 13B0    C9          RET
 94 13B1    E5 BDV1:    PUSH    H       ;SAVE DIVIDEND
 95 13B2    26          MVI     H,0     ;CLEAR COUNTER
    13B3    00
 96 13B4    7A          MOV     A,D
 97 13B5    24 BDV2:    INR     H       ;INC. COUNTER
 98 13B6    E6          ANI     80
    13B7    80
 99 13B8    C2          JNZ     BDV3    ;IF M.S.B.=1
    13B9    C5
    13BA    13
100 13BB    AF          XRA     A       ;SHIFT
101 13BC    7B          MOV     A,E
102 13BD    17          RAL
103 13BE    5F          MOV     E,A
104 13BF    7A          MOV     A,D     ;  DE
105 13C0    17          RAL
106 13C1    57          MOV     D,A     ;  LEFT
107 13C2    C3          JMP     BDV2
    13C3    B5
    13C4    13
108 13C5    E3 BDV3:    XTHL            ;RESTORE DIVIDEND
109 13C6    CD          CALL    BSU     ;HL-DE=HL
    13C7    5E
    13C8    13
110 13C9    DA          JC      BDV4    ;IF -VE
    13CA    D0
    13CB    13
111 13CC    0C          INR     C       ;SET L.S.B. OF QUOTIENT
112 13CD    C3          JMP     BDV5
    13CE    D1
    13CF    13
113 13D0    19 BDV4:    DAD     D
114 13D1    E3 BDV5:    XTHL            ;SAVE DIVIDEND
115 13D2    25          DCR     H
116 13D3    CA          JZ      BDV6    ;IF COUNTER=0
    13D4    E6
    13D5    13
117 13D6    AF          XRA     A       ;SHIFT
118 13D7    79          MOV     A,C
119 13D8    17          RAL
120 13D9    4F          MOV     C,A     ;  BC
121 13DA    78          MOV     A,B
122 13DB    17          RAL
```

```
123 13DC    47          MOV     B,A     ;  LEFT
124 13DD    7A          MOV     A,D     ;SHIFT
125 13DE    1F          RAR
126 13DF    57          MOV     D,A
127 13E0    7B          MOV     A,E     ;  DE
128 13E1    1F          RAR
129 13E2    5F          MOV     E,A     ;  RIGHT
130 13E3    C3          JMP     BDV3    ;NEXT BIT
    13E4    C5
    13E5    13
131 13E6    E1  BDV6:   POP     H       ;RESTORE REMAINDER
132 13E7    AF          XRA     A       ;ROUND
133 13E8    7D          MOV     A,L
134 13E9    17          RAL
135 13EA    6F          MOV     L,A     ;  OFF
136 13EB    7C          MOV     A,H
137 13EC    17          RAL
138 13ED    67          MOV     H,A     ;  THE
139 13EE    BA          CMP     D
140 13EF    DA          JC      NCY0
    13F0    FB
    13F1    13
141 13F2    C2          JNZ     CY0     ;  LAST
    13F3    FA
    13F4    13
142 13F5    7D          MOV     A,L
143 13F6    BB          CMP     E
144 13F7    DA          JC      NCY0    ;  BIT
    13F8    FD
    13F9    13
145 13FA    03  CY0:    INX     B
146 13FB    60  NCY0:   MOV     H,B
147 13FC    69          MOV     L,C
148 13FD    C1          POP     B
149 13FE    C9          RET
150             .END
```

AD1	128B	AD2	128D	AD3	1297	AD4	1299
ADDSR	1282	BDV	139F	BDV1	13B1	BDV2	13B5
BDV3	13C5	BDV4	13D0	BDV5	13D1	BDV6	13E6
BMU	1365	BSU	135E	CEXP	133B	CY0	13FA
DEND	122A	DV1	11BD	DV10	127E	DV2	11C3
DV3	11EB	DV4	11F9	DV5	1210	DV6	123A
DV7	124A	DV8	1263	DV9	1272	DVD	11AE
FSTZ	10A2	IFZ	1098	MI	130F	MT	1311
MT1	1318	MU1	10CE	MU2	10D7	MU3	10EF
MU4	10FB	MU5	1165	MU6	1172	MU7	117F
MU8	1194	MU9	11A0	MUPY	10C1	MZ	1391
NAC1	1346	NAC2	1351	NAC3	135D	NCY1	1048
NCY2	12E4	NCY0	13FB	NOK	137E	OVFM	1399
RBCF	1302	RBCR	12E6	RDEL	12F4	RDER	12ED
RESU	1206	RET1	107D	REXP	1352	RFOR	132D
RHLL	12FB	RI	1323	ROPD	12C0	SB1	12AA
SB2	12AC	SB3	12BC	SB4	12BE	SBM	136F
SM1	102E	SM3	104C	SM4	105C	SM5	1069
SMI1	1018	SUBB	1081	SUBSR	129B	SUMM	1003
TBM	11A9	TRM	11AA				

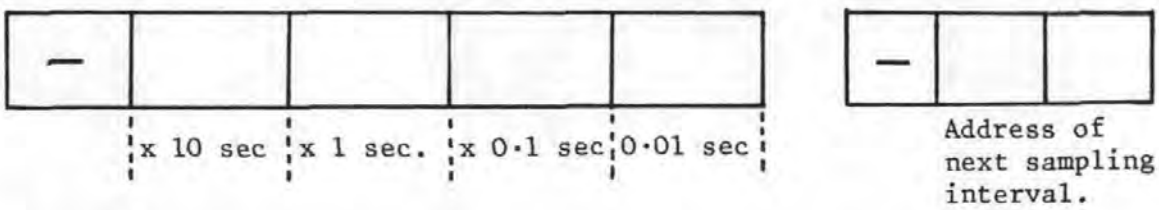
*

APPENDIX 7.

Sampling Rate and Data Logger Programming.

(A) Sampling Interval Programming .

The sampling rate can be programmed to a sequence of different intervals. The location of the first sampling itnerval is in constant address '00'. The setting up of the sampling interval is by loading the constant table as follows:

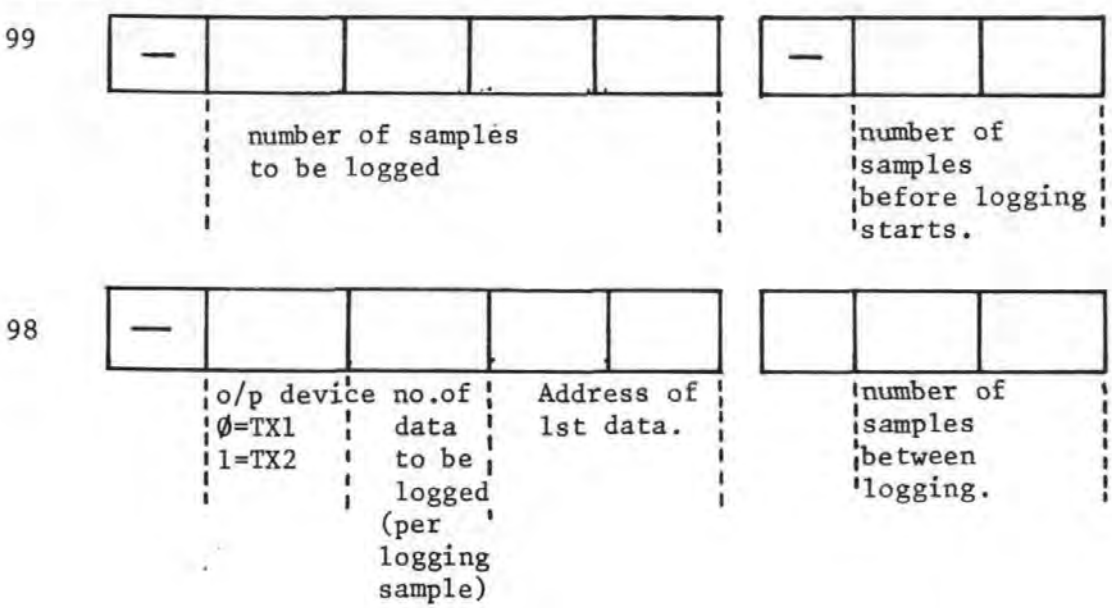


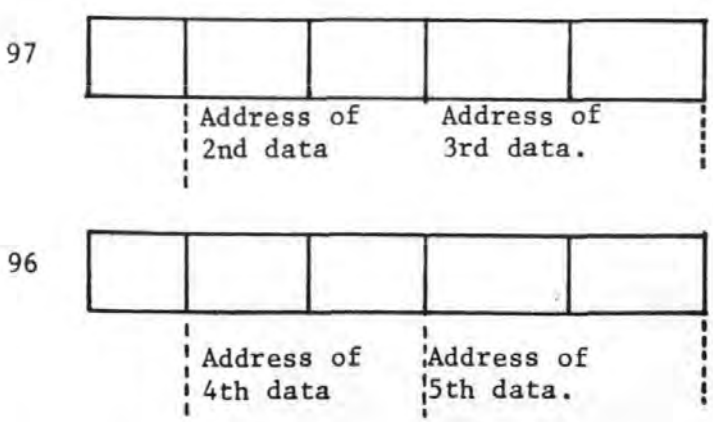
N.B. The shortest sampling interval is 0·05 sec.

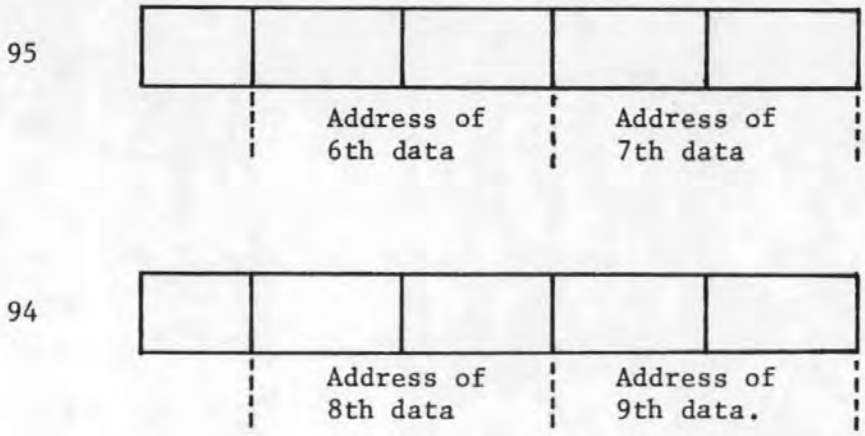
(B) Data Logger Programming.

The Data Logger can be programed by loading the constant table as follows.

Constant Addr.







This Data Logger can be logged up to nine data per sample. Its output through communication interface TX1 or TX2 is normally in the form of two start bits, followed by eight bit ASC111 and then one stop bit. The band rate is 110. These can be changed by reprogramming the programmable timer and the corresponding communication interface. The print out of the data is in the following pattern.

Sample no. 1st Data 2nd Data 3rd Data 4th Data 5th Data 6th Data

7th Data 8th Data 9th Data

The data is presented in the form of a mantissa and an exponent (e.g. -.0001 E-00).

APPENDIX 8

Listing of Pseudo-random Binary Sequence Program and Maximum Cycle Length Test Program.

```
 1                      .TITLE 'CHECK FB. FOR MAX. CYCLE LH.'
 2                      .HEX
 3  1900     00         .ORG 1900
 4                      ECHO=0F71
 5  1900     1E         MVI     E,0       ;CLEAR FEEDBACK COUNTER
    1901     00
 6  1902     16         MVI     D,080     ;LOAD FEEDBACK MASK
    1903     80
 7  1904     1C ALTR:   INR     E
 8  1905     7A         MOV     A,D
 9  1906     07         RLC
10  1907     57         MOV     D,A
11  1908     06         MVI     B,0
    1909     00
12  190A     48         MOV     C,B
13  190B     26         MVI     H,0FF
    190C     FF
14  190D     37         STC
15  190E     03 FDB:    INX     B
16  190F     7C         MOV     A,H
17  1910     1F         RAR
18  1911     67         MOV     H,A
19  1912     DA         JC      LONE
    1913     1D
    1914     19
20  1915     A2         ANA     D
21  1916     CA         JZ      FDB
    1917     0E
    1918     19
22  1919     37         STC
23  191A     C3         JMP     ETST
    191B     22
    191C     19
24  191D     A2 LONE:   ANA     D
25  191E     C2         JNZ     FDB
    191F     0E
    1920     19
26  1921     37         STC
27  1922     7C ETST:   MOV     A,H
28  1923     FE         CPI     0FF
    1924     FF
29  1925     C2         JNZ     FDB
    1926     0E
    1927     19
30  1928     78         MOV     A,B
31  1929     FE         CPI     01
    192A     01
32  192B     C2         JNZ     ALTR
    192C     04
    192D     19
33  192E     79         MOV     A,C
34  192F     FE         CPI     0FF
    1930     FF
35  1931     C2         JNZ     ALTR
    1932     04
    1933     19
36  1934     7B         MOV     A,E
37  1935     F6         ORI     30
    1936     30
38  1937     CD         CALL    ECHO
    1938     71
    1939     0F
39  193A     C3         JMP     0499
    193B     99
    193C     04
40                      .END
ALTR  1904    ECHO  0F71    ETST  1922    FDB  190E
```

```
*TTY:<5,75:OPEL

PSEUDO-RANDOM BIN. SEQ. GEN.          CALOS-80 V2.12 10/13/70 PAGE    1

   1                .TITLE 'PSEUDO-RANDOM BIN. SEQ. GEN.'
   2                .HEX
   3 1400     00 .ORG 1400
   4 1400     2A          LHLD    0824     ;LOAD
     1401     24
     1402     08
   5 1403     E5          PUSH    H        ;   PREVIOUS
   6 1404     F1          POP     PSW      ;   O/P
   7 1405     CD          CALL    PRBS
     1406     1E
     1407     14
   8 1408     F5          PUSH    PSW      ;SAVE
   9 1409     E1          POP     H        ;   REG.
  10 140A     22          SHLD    0824     ;   STATE
     140B     24
     140C     08
  11 140D     3A          LDA     0821     ;FETCH S.S. SETTING
     140E     21
     140F     08
  12 1410     DA          JC      ATR
     1411     18
     1412     14
  13 1413     D6          SUI     0C       <- 18
     1414     0C
  14 1415     C3          JMP     OPR
     1416     1A
     1417     14
  15 1418     C6 ATR:     ADI     0C       <- 19
     1419     0C
  16 141A     32 OPR:     STA     0803     ;O/P TO PROCESS
     141B     03
     141C     08
  17 141D     C9          RET
  18 141E     1F PRBS:    RAR
  19 141F     47          MOV     B,A
  20 1420     DA          JC      HG
     1421     2E
     1422     14
  21 1423     FE          CPI     0
     1424     00
  22 1425     CA          JZ      NDEN
     1426     2C
     1427     14
  23 1428     E6          ANI     08
     1429     08
  24 142A     78          MOV     A,B
  25 142B     C8          RZ
  26 142C     37 NDEN:    STC
  27 142D     C9          RET
  28 142E     E6 HG:      ANI     08
     142F     08
  29 1430     78          MOV     A,B
  30 1431     C0          RNZ
  31 1432     37          STC
  32 1433     C9          RET
  33                .END
ATR     1418      HG     142E      NDEN    142C      OPR     141A

PRBS    141E
```

APPENDIX 9

Listing of Cross-correlation and Auto-correlation Program.

```
 1                .TITLE "CROSS & AUTO-CORRELATION PROGRAM"
 2                .HEX
 3                ;THIS PROGRAM READS FROM THE DATA LOGGER OUTPUT IN PAPER
 4                ;TAPE FORM WITH I/P(HEATER) FIRST AND O/P(LEVEL) SECOND.
 5                ;THEN IT CALCULATES THE CROSS-CORRELATION FUNCTION AND
 6                ;THE AUTO-CORRELATION FUNCTION AND OUTPUT THEM IN
 7                ;GRAPH FORM.
 8                BDV=139F
 9                DRD=0FC3
10                ECHO=0F6A
11                INAD2=0F1C
12                OPSA=0F7B
13                SRIP2=015C
14                TREN=049D
15                TYSP=0FDC
16 1400    00  .ORG 1400
17 1400    3A          LDA     0BFC    ;ENABLE READER
   1401    FC
   1402    0B
18 1403    F6          ORI     02
   1404    02
19 1405    32          STA     0BFC
   1406    FC
   1407    0B
20 1408    32          STA     8011
   1409    11
   140A    80
21 140B    3E          MVI     A,14    ;SET INITIAL CONDN. FOR
   140C    14
22 140D    32          STA     0A1A    ;   FILTERS
   140E    1A
   140F    0A
23 1410    32          STA     0A1D
   1411    1D
   1412    0A
24 1413    21          LXI     H,0028  ;SET X FOR LEVEL
   1414    28
   1415    00
25 1416    22          SHLD    0A1B
   1417    1B
   1418    0A
26 1419    21          LXI     H,00F0  ;SET X FOR HEATER
   141A    F0
   141B    00
27 141C    22          SHLD    0A1E
   141D    1E
   141E    0A
28 141F    11          LXI     D,1C00
   1420    00
   1421    1C
29 1422    21          LXI     H,1E00
   1423    00
   1424    1E
30 1425    CD COR1:    CALL    SRIP2
   1426    5C
   1427    01
31 1428    FE          CPI     0       ;IF =0
   1429    00
32 142A    CA          JZ      STOP
   142B    75
   142C    14
33 142D    FE          CPI     0D      ;IF CR
   142E    0D
34 142F    C2          JNZ     COR1
   1430    25
   1431    14
35 1432    CD          CALL    ECHO
   1433    6A
   1434    0F
36 1435    CD DOT1:    CALL    SRIP2
   1436    5C
   1437    01
37 1438    FE          CPI     0
   1439    00
38 143A    CA          JZ      STOP
   143B    75
   143C    14
39 143D    FE          CPI     '.'     ;IF '.'
   143E    2E
40 143F    C2          JNZ     DOT1
   1440    35
   1441    14
41 1442    CD          CALL    SRIP2
   1443    5C
   1444    01
42 1445    CD          CALL    SRIP2
   1446    5C
   1447    01
43 1448    D5          PUSH    D
```

```
44 1448    CD          CALL    INAD2   ;READ HEATER I/P
   1449    1C
   144A    0F
45 144B    4B          MOV     C,E
46 144C    D1          POP     D
47 144D    CD DOT2:    CALL    SRIP2
   144E    5C
   144F    01
48 1451    FE          CPI     '.'     ;IF '.'
   1452    2E
49 1453    C2          JNZ     DOT2
   1454    4E
   1455    14
50 1456    CD          CALL    SRIP2
   1457    5C
   1458    01
51 1459    CD          CALL    SRIP2
   145A    5C
   145B    01
52 145C    D5          PUSH    D
53 145D    CD          CALL    INAD2   ;READ LEVEL O/P
   145E    1C
   145F    0F
54 1460    7B          MOV     A,E
55 1461    D1          POP     D
56 1462    2F          CMA             ;CONDN. O/P
57 1463    CD          CALL    HPL
   1464    F4
   1465    15
58 1466    12          STAX    D       ;STORE O/P
59 1467    79          MOV     A,C
60 1468    0F          RRC             ;CONDN. I/P
61 1469    0F          RRC
62 146A    E6          ANI     3F
   146B    3F
63 146C    CD          CALL    HPH
   146D    1E
   146E    16
64 146F    77          MOV     M,A     ;STORE I/P
65 1470    13          INX     D
66 1471    23          INX     H
67 1472    C3          JMP     COR1
   1473    25
   1474    14
68 1475    CD STOP:    CALL    DRD     ;DISABLE READER
   1476    C3
   1477    0F
69 1478    22          SHLD    0A16
   1479    16
   147A    0A
70 147B    EB          XCHG
71 147C    22          SHLD    0A14
   147D    14
   147E    0A
72 147F    CD          CALL    DOTS
   1480    E6
   1481    15
73 1482    CD          CALL    DOTS
   1483    E6
   1484    15
74 1485    3E          MVI     A,'!'
   1486    21
75 1487    CD          CALL    ECHO
   1488    6A
   1489    0F
76 148A    CD          CALL    DOTS
   148B    E6
   148C    15
77 148D    3E          MVI     A,0D
   148E    0D
78 148F    CD          CALL    ECHO
   1490    6A
   1491    0F
79 1492    06          MVI     B,0
   1493    00
80 1494    11 COM:     LXI     D,1C00
   1495    00
   1496    1C
81 1497    21          LXI     H,1E00
   1498    00
   1499    1E
82 149A    78          MOV     A,B
83 149B    83          ADD     E
84 149C    5F          MOV     E,A     ;E=B+E
85 149D    AF          XRA     A
86 149E    4F          MOV     C,A
87 149F    32          STA     0A18
   14A0    18
   14A1    0A
88 14A2    32          STA     0A19
   14A3    19
   14A4    0A
89 14A5    C5 CRCL:    PUSH    B       ;CROSS-CORRELATION CALN.
90 14A6    7E          MOV     A,M
```

```
 91 14A7    E6          ANI     7F      ;DELETE SIGN BIT
    14A8    7F
 92 14A9    47          MOV     B,A
 93 14AA    1A          LDAX    D
 94 14AB    E6          ANI     7F      ;DELETE SIGN BIT
    14AC    7F
 95 14AD    4F          MOV     C,A
 96 14AE    0C          INR     C
 97 14AF    AF          XRA     A
 98 14B0    0D ML:      DCR     C       ;(DE)*(HL)
 99 14B1    CA          JZ      SM
    14B2    B8
    14B3    14
100 14B4    80          ADD     B
101 14B5    C3          JMP     ML
    14B6    B0
    14B7    14
102 14B8    06 SM:      MVI     B,0
    14B9    00
103 14BA    4F          MOV     C,A
104 14BB    1A          LDAX    D
105 14BC    AE          XRA     M
106 14BD    E6          ANI     80      ;EXTRACT SIGN BIT
    14BE    80
107 14BF    CA          JZ      SM1     ;IF BOTH SIGN THE SAME
    14C0    CC
    14C1    14
108 14C2    79          MOV     A,C     ;   THEN 2'S COMPLEMENT PRODUCT
109 14C3    2F          CMA
110 14C4    C6          ADI     01
    14C5    01
111 14C6    4F          MOV     C,A
112 14C7    3E          MVI     A,0
    14C8    00
113 14C9    CE          ACI     0FF
    14CA    FF
114 14CB    47          MOV     B,A
115 14CC    E5 SM1:     PUSH    H
116 14CD    2A          LHLD    0A18
    14CE    18
    14CF    0A
117 14D0    09          DAD     B
118 14D1    22          SHLD    0A18
    14D2    18
    14D3    0A
119 14D4    E1          POP     H
120 14D5    13          INX     D
121 14D6    23          INX     H
122 14D7    C1          POP     B
123 14D8    0C          INR     C
124 14D9    3A          LDA     0A14
    14DA    14
    14DB    0A
125 14DC    D6          SUI     15      ;TERMINATE AT 21 POINTS BEFORE E
ND
    14DD    15
126 14DE    BD          CMP     L
127 14DF    D2          JNC     CRCL
    14E0    A5
    14E1    14
128 14E2    2A          LHLD    0A18
    14E3    18
    14E4    0A
129 14E5    CD          CALL    OPSA    ;O/P RESULT
    14E6    7B
    14E7    0F
130 14E8    79          MOV     A,C     ;NO.OF DATA/ 8
131 14E9    0F          RRC
132 14EA    0F          RRC
133 14EB    0F          RRC
134 14EC    E6          ANI     1F
    14ED    1F
135 14EE    5F          MOV     E,A
136 14EF    16          MVI     D,0
    14F0    00
137 14F1    3E          MVI     A,80    ;IF
    14F2    80
138 14F3    A4          ANA     H       ;  O/P
139 14F4    CA          JZ      DIV     ;  NOT -VE
    14F5    0B
    14F6    15
140 14F7    7C          MOV     A,H     ;2'S COMPLEMENT HL
141 14F8    2F          CMA
142 14F9    67          MOV     H,A
143 14FA    7D          MOV     A,L
144 14FB    2F          CMA
145 14FC    6F          MOV     L,A
146 14FD    23          INX     H
147 14FE    CD          CALL    BDV
    14FF    9F
    1500    13
148 1501    7C          MOV     A,H
149 1502    2F          CMA
```

```
150 1503    67          MOV     H,A
151 1504    79          MOV     A,L
152 1505    2F          CMA
153 1506    6F          MOV     L,A
154 1507    23          INX     H
155 1508    C3          JMP     SG
    1509    CE
    150A    15
156 150B    CD DIV:     CALL    BDV
    150C    9F
    150D    13
157 150E    1E SG:      MVI     E,18    ;ADD 24 TO LIFT THE GRAPH UP
    150F    18
158 1510    16          MVI     D,0
    1511    00
159 1512    19          DAD     D
160 1513    7C          MOV     A,H
161 1514    E6          ANI     80
    1515    80
162 1516    C2          JNZ     AST
    1517    24
    1518    15
163 1519    23          INX     H
164 151A    2D PLOT:    DCR     L
165 151B    CA          JZ      AST
    151C    24
    151D    15
166 151E    CD          CALL    TYSP
    151F    DC
    1520    0F
167 1521    C3          JMP     PLOT
    1522    1A
    1523    15
168 1524    3E AST:     MVI     A,'*'
    1525    2A
169 1526    CD          CALL    ECHO
    1527    6A
    1528    0F
170 1529    3E          MVI     A,0D    ;TYPE CR
    152A    0D
171 152B    CD          CALL    ECHO
    152C    6A
    152D    0F
172 152E    04          INR     B
173 152F    78          MOV     A,B
174 1530    FE          CPI     14
    1531    14
175 1532    C2          JNZ     COM
    1533    94
    1534    14
176                 ;AUTO-CORRELATION
177 1535    CD          CALL    DOTS
    1536    E6
    1537    15
178 1538    3E          MVI     A,'!'
    1539    21
179 153A    CD          CALL    ECHO
    153B    6A
    153C    0F
180 153D    CD          CALL    DOTS
    153E    E6
    153F    15
181 1540    3E          MVI     A,0D
    1541    0D
182 1542    CD          CALL    ECHO
    1543    6A
    1544    0F
183 1545    06          MVI     B,0
    1546    00
184 1547    21 AUCO:    LXI     H,1E00
    1548    00
    1549    1E
185 154A    AF          XRA     A
186 154B    32          STA     0A18    ;CLEAR SP & C
    154C    18
    154D    0A
187 154E    32          STA     0A19
    154F    19
    1550    0A
188 1551    4F          MOV     C,A
189 1552    58          MOV     E,B
190 1553    54          MOV     D,H
191 1554    C5 CAL:     PUSH    B
192 1555    7E          MOV     A,M     ;CONDN. I/P
193 1556    E6          ANI     7F      ;DELET SIGN BIT
    1557    7F
194 1558    47          MOV     B,A     ;B=I/P
195 1559    1A          LDAX    D
196 155A    E6          ANI     7F      ;DELET SIGN BIT
    155B    7F
197 155C    4F          MOV     C,A     ;C=SHIFTED I/P
198 155D    0C          INR     C
199 155E    AF          XRA     A
```

```
 200 155F    0D MP:    DCR     C       ;B*C=A
 201 1560    CA        JZ      SMM
     1561    67
     1562    15
 202 1563    80        ADD     B
 203 1564    C3        JMP     MP
     1565    5F
     1566    15
 204 1567    06 SMM:   MVI     B,0     ;SUMMATION
     1568    00
 205 1569    4F        MOV     C,A
 206 156A    1A        LDAX    D
 207 156B    AE        XRA     M
 208 156C    E6        ANI     80      ;EXTRACT SIGN
     156D    80
 209 156E    CA        JZ      SMM1    ;IF BOTH SIGN THE SAME
     156F    7B
     1570    15
 210 1571    79        MOV     A,C     ;   THEN 2'S COMPLEMENT PRODUCT
 211 1572    2F        CMA
 212 1573    C6        ADI     01
     1574    01
 213 1575    4F        MOV     C,A
 214 1576    3E        MVI     A,0
     1577    00
 215 1578    CE        ACI     OFF
     1579    FF
 216 157A    47        MOV     B,A
 217 157B    E5 SMM1:  PUSH    H
 218 157C    2A        LHLD    0A18
     157D    18
     157E    0A
 219 157F    09        DAD     B
 220 1580    22        SHLD    0A18
     1581    18
     1582    0A
 221 1583    E1        POP     H
 222 1584    23        INX     H
 223 1585    13        INX     D
 224 1586    C1        POP     B
 225 1587    0C        INR     C
 226 1588    3A        LDA     0A16
     1589    16
     158A    0A
 227 158B    D6        SUI     15      ;TERMINATE AT 15 POINTS BEFORE E
ND
     158C    15
 228 158D    BD        CMP     L
 229 158E    D2        JNC     CAL
     158F    54
     1590    15
 230 1591    2A        LHLD    0A18    ;SP/C=HL
     1592    18
     1593    0A
 231 1594    CD        CALL    OPSA    ;O/P RESULT
     1595    7B
     1596    0F
 232 1597    79        MOV     A,C     ;NO. OF DATA/ 4
 233 1598    0F        RRC
 234 1599    0F        RRC
 235 159A    E6        ANI     3F
     159B    3F
 236 159C    5F        MOV     E,A
 237 159D    16        MVI     D,0
     159E    00
 238 159F    3E        MVI     A,80
     15A0    80
 239 15A1    A4        ANA     H
 240 15A2    CA        JZ      DIVE
     15A3    B9
     15A4    15
 241 15A5    7C        MOV     A,H
 242 15A6    2F        CMA
 243 15A7    67        MOV     H,A     ;2'S COMPLEMENT HL
 244 15A8    7D        MOV     A,L
 245 15A9    2F        CMA
 246 15AA    6F        MOV     L,A
 247 15AB    23        INX     H
 248 15AC    CD        CALL    BDV
     15AD    9F
     15AE    13
 249 15AF    7C        MOV     A,H
 250 15B0    2F        CMA
 251 15B1    67        MOV     H,A
 252 15B2    7D        MOV     A,L
 253 15B3    2F        CMA
 254 15B4    6F        MOV     L,A
 255 15B5    23        INX     H
 256 15B6    C3        JMP     SGA
     15B7    BC
     15B8    15
 257 15B9    CD DIVE:  CALL    BDV
     15BA    9F
     15BB    13
```

```
259 15DC  1E 55A:   MVI   E,0A
    15DD  0A
259 15DE  16        MVI   D,0
    15DF  00
260 15E0  19        DAD   D         ;ADD TEN TO LIFT THE GRAHP UP.
261 15E1  7C        MOV   A,H
262 15E2  E6        ANI   80
    15E3  80
263 15E4  C2        JNZ   STR
    15E5  D2
    15E6  15
264 15E7  23        INX   H
265 15E8  2D GRH:   DCR   L
266 15E9  CA        JZ    STR
    15EA  D2
    15EB  15
267 15EC  CD        CALL  TYSP
    15ED  DC
    15EE  0F
268 15EF  C3        JMP   GRH
    15F0  C8
    15F1  15
269 15F2  3E STR:   MVI   A,'*'
    15F3  2A
270 15F4  CD        CALL  ECHO
    15F5  6A
    15F6  0F
271 15F7  3E        MVI   A,0D
    15F8  0D
272 15F9  CD        CALL  ECHO    ;TYPE CR & LF
    15FA  6A
    15FB  0F
273 15FC  04        INR   B
274 15FD  78        MOV   A,B
275 15FE  FE        CPI   14
    15FF  14
276 1600  C2        JNZ   AUCO
    1601  47
    1602  15
277 1603  C3        JMP   TREN
    1604  9D
    1605  04
```



```
  1             ;SUBROUTINE FOR TYPING 14 DOTS
  2 15E6  C5 DOTS:  PUSH  B
  3 15E7  06        MVI   B,0E
    15E8  0E
  4 15E9  3E DT1:   MVI   A,'.'
    15EA  2E
  5 15EB  CD        CALL  ECHO
    15EC  6A
    15ED  0F
  6 15EE  05        DCR   B
  7 15EF  C2        JNZ   DT1
    15F0  E9
    15F1  15
  8 15F2  C1        POP   B
  9 15F3  C9        RET
 10             ;THIS SUBROUTINE PERFORMS THE HIGH PASS FILTER
 11             ;FUNCTION FOR THE LEVEL.
 12 15F4  D5 HPL:  PUSH  D
 13 15F5  E5       PUSH  H
 14 15F6  F5       PUSH  PSW
 15 15F7  3A       LDA   0A1A
    15F8  1A
    15F9  0A
 16 15FA  5F       MOV   E,A
 17 15FB  16       MVI   D,0
    15FC  00
 18 15FD  2A       LHLD  0A1B
    15FE  1B
    15FF  0A
 19 1600  CD       CALL  BDV     ;X(K) / T = X
    1601  9F
    1602  13
 20 1603  F1       POP   PSW
 21 1604  95       SUB   L       ;U(K+1) - X = Y(K+1)
 22 1605  F5       PUSH  PSW
 23 1606  2A       LHLD  0A1B    ;UPDATE X(K)
    1607  1B
    1608  0A
 24 1609  5F       MOV   E,A     ;   I.E. X(K+1) = X(K) + Y(K+1)
 25 160A  D2       JNC   HL1
    160B  0F
    160C  16
 26 160D  16       MVI   D,0FF
    160E  FF
 27 160F  19 HL1:  DAD   D
 28 1610  22       SHLD  0A1B
    1611  1B
    1612  0A
```

```
 29 1613    F1          POP     PSW
 30 1614    F2          JP      HL2
    1615    1D
    1616    16
 31 1617    2F          CMA
 32 1618    3C          INR     A
 33 1619    F6          ORI     80
    161A    80
 34 161B    E1 HL2:     POP     H
 35 161C    D1          POP     D
 36 161D    C9          RET
 37                  ;THIS SUBROUTINE PERFORMS THE HIGH PASS FILTER FUNCTION
 38                  ;FOR THE HEATER.
 39 161E    D5 HPH:     PUSH    D
 40 161F    E5          PUSH    H
 41 1620    F5          PUSH    PSW
 42 1621    3A          LDA     0A1D
    1622    1D
    1623    0A
 43 1624    5F          MOV     E,A
 44 1625    16          MVI     D,0
    1626    00
 45 1627    2A          LHLD    0A1E
    1628    1E
    1629    0A
 46 162A    CD          CALL    BDV     ;X(K) / T = X
    162B    9F
    162C    13
 47 162D    F1          POP     PSW
 48 162E    95          SUB     L       ;U(K+1) - X = Y(K+1)
 49 162F    F5          PUSH    PSW
 50 1630    2A          LHLD    0A1E    ;UPDATE X(K)
    1631    1E
    1632    0A
 51 1633    5F          MOV     E,A     ;   I.E. X(K+1) = X(K) + Y(K+1)
 52 1634    D2          JNC     HH1
    1635    39
    1636    16
 53 1637    16          MVI     D,0FF
    1638    FF
 54 1639    19 HH1:     DAD     D
 55 163A    22          SHLD    0A1E
    163B    1E
    163C    0A
 56 163D    F1          POP     PSW
 57 163E    F2          JP      HH2
    163F    45
    1640    16
 58 1641    2F          CMA
 59 1642    3C          INR     A
 60 1643    F6          ORI     80
    1644    80
 61 1645    E1 HH2:     POP     H
 62 1646    D1          POP     D
 63 1647    C9          RET
 64                  .END
```

AST	1524	AUC0	1547	BDV	139F	CAL	1554
COM	1494	COR1	1425	CRCL	14A5	DIV	150B
DIVE	15B9	DOT1	1435	DOT2	144E	DOTS	15E6
DRD	0FC3	DT1	15E9	ECHO	0F6A	GRH	15C8
HH1	1639	HH2	1645	HL1	160F	HL2	161B
HPH	161E	HPL	15F4	INAD2	0F1C	ML	14B0
MP	155F	OPSA	0F7B	PLOT	151A	SG	150E
SGA	15BC	SM	14B8	SM1	14CC	SMM	1567
SMM1	157B	SRIP2	015C	STOP	1475	STR	15D2
TREN	049D	TYSP	0FDC				

*

APPENDIX 10

Listing of Three-term Controller Program.

```
                .TITLE "3-TERM CONTROLLER ALGORITHM"
                .HEX
                SUMM=1003
                MUPY=10C1
                DVD=11AE
                RDER=12ED
1C00    00      .ORG 1C00
1C00    3A      SUSB:   LDA  0808    ;FETCH I/P STATUS
1C01    08
1C02    08
1C03    E6              ANI  01
1C04    01
1C05    CA              JZ   EXPF    ;IF B.C.D.
1C06    24
1C07    1C
1C08    06              MVI  B,0     ;START BINARY TO BCD CONVERSION
1C09    00
1C0A    21      SUSB:   LXI  H,0806
1C0B    06
1C0C    08
1C0D    7E              MOV  A,M     ;FETCH I/P
1C0E    FE              CPI  0A
1C0F    0A
1C10    DA              JC   STHA    ;IF < 10
1C11    1D
1C12    1C
1C13    D6              SUI  0A
1C14    0A
1C15    04              INR  B
1C16    77              MOV  M,A     ;STORE REMAINDER
1C17    78              MOV  A,B
1C18    27              DAA          ;B.C.D. CORRECTION OF QUOTIENT
1C19    47              MOV  B,A     ;SAVE QUOTIENT
1C1A    C3              JMP  SUSB
1C1B    0A
1C1C    1C
1C1D    07      STHA:   RLC
1C1E    07              RLC
1C1F    07              RLC
1C20    07              RLC
1C21    77              MOV  M,A     ;PLACE L.S.D.
1C22    23              INX  H
1C23    70              MOV  M,B     ;PLACE M.S. 2 DIGIT
                ;EXPONENT FILTER Y(T+1)=(1-K)*Y(T)+K*U(T+1)
1C24    2A      EXPF:   LHLD 81E     ;FETCH
1C25    1E
1C26    08
1C27    EB              XCHG
1C28    3A              LDA  820     ;-K
1C29    20
1C2A    08
1C2B    E6              ANI  7F
1C2C    7F
1C2D    6F              MOV  L,A
1C2E    01              LXI  B,1000  ;LOAD 1
1C2F    00
1C30    10
1C31    26              MVI  H,0C1   ;   TO BC & H
1C32    C1
1C33    CD              CALL SUMM    ;1-K
1C34    03
1C35    10
1C36    3A              LDA  821     ;FETCH
1C37    21
1C38    08
1C39    4F              MOV  C,A     ;   Y(T)
1C3A    3A              LDA  822
1C3B    22
1C3C    08
1C3D    47              MOV  B,A
1C3E    3A              LDA  823
1C3F    23
1C40    08
1C41    67              MOV  H,A
1C42    CD              CALL MUPY    ;(1-K)*Y(T)
1C43    C1
1C44    10
1C45    7D              MOV  A,L     ;STORE RESULT
1C46    32              STA  823
1C47    23
1C48    08
1C49    EB              XCHG
1C4A    22              SHLD 821     ;   IN '11'
1C4B    21
1C4C    08
1C4D    2A              LHLD 81E     ;FETCH
1C4E    1E
1C4F    08
1C50    EB              XCHG
1C51    2A              LHLD 820     ;   K
1C52    20
1C53    08
```

```
53 1C84    3A          LDA  806      ;FETCH
   1C85    06
   1C86    08
54 1C87    4F          MOV  C,A
55 1C88    3A          LDA  807      ;   U(T+1)
   1C89    07
   1C8A    08
56 1C8B    47          MOV  B,A
57 1C8C    26          MVI  H,0C0    ;   FROM '3'
   1C8D    C0
58 1C8E    CD          CALL MUPY     ;K*U(T+1)
   1C8F    C1
   1C90    10
59 1C91    3A          LDA  821      ;FETCH
   1C92    21
   1C93    08
60 1C94    4F          MOV  C,A
61 1C95    3A          LDA  822      ;   (1-K)*Y(T)
   1C96    22
   1C97    08
62 1C98    47          MOV  B,A
63 1C99    3A          LDA  823      ;   FROM '11'
   1C9A    23
   1C9B    08
64 1C9C    67          MOV  H,A
65 1C9D    CD          CALL SUMM     ;Y(T+1)=(1-K)*Y(T)+K*U(T+1)
   1C9E    03
   1C9F    10
66 1CA0    7D          MOV  A,L      ;STORE RESULT
67 1CA1    32          STA  823
   1CA2    23
   1CA3    08
68 1CA4    EB          XCHG
69 1CA5    22          SHLD 821      ;   IN '11'
   1CA6    21
   1CA7    08
70 1CA8    01 BCDI:    LXI  B,815    ;SHIFT PREVIOUS
   1CA9    15
   1CAA    08
71 1CAB    11          LXI  D,818    ;       ERRORS
   1CAC    18
   1CAD    08
72 1CAE    21          LXI  H,81B    ;       E(T-1)
   1CAF    1B
   1CB0    08
73 1CB1    3E          MVI  A,03
   1CB2    03
74 1CB3    F5 MVA:     PUSH PSW
75 1CB4    1A          LDAX D        ;       E(T-2)
76 1CB5    77          MOV  M,A
77 1CB6    0A          LDAX B
78 1CB7    12          STAX D
79 1CB8    03          INX  B
80 1CB9    13          INX  D
81 1CBA    23          INX  H
82 1CBB    F1          POP  PSW
83 1CBC    3D          DCR  A
84 1CBD    C2          JNZ  MVA
   1CBE    83
   1CBF    1C
85                ;CALCULATE E(T)
86 1C90    2A          LHLD 821      ;FETCH MEASUREMENT
   1C91    21
   1C92    08
87 1C93    44          MOV  B,H      ;SAVE I/P IN BC
88 1C94    7D          MOV  A,L
89 1C95    E6          ANI  0F0
   1C96    F0
90 1C97    4F          MOV  C,A
91 1C98    2A          LHLD 809      ;FETCH SET POINT
   1C99    09
   1C9A    08
92 1C9B    54          MOV  D,H      ;SAVE
93 1C9C    7D          MOV  A,L      ;       SET POINT
94 1C9D    E6          ANI  0F0      ;       IN
   1C9E    F0
95 1C9F    5F          MOV  E,A      ;       DE
96 1CA0    26          MVI  H,40     ;SET MEASUREMENT -VE
   1CA1    40
97 1CA2    2E          MVI  L,0C0    ;SET SET POINT +VE
   1CA3    C0
98 1CA4    CD          CALL SUMM
   1CA5    03
   1CA6    10
99 1CA7    EB          XCHG
100 1CA8   22          SHLD 0815     ;STORE E(T)
   1CA9    15
   1CAA    08
101 1CAB   7B          MOV  A,E
102 1CAC   32          STA  0817
   1CAD    17
   1CAE    08
```

```
103 1CAF   2A        LHLD 0800    ;FETCH TS SAMPLING INTV.
    1CB0   00
    1CB1   08
104 1CB2   44        MOV  B,H
105 1CB3   4D        MOV  C,L
106 1CB4   26        MVI  H,0C2   ;       ONTO BC
    1CB5   C2
107 1CB6   CD        CALL COPR    ;SHIFT OPERATOR.
    1CB7   20
    1CB8   1E
108 1CB9   E5        PUSH H       ;FETCH
109 1CBA   21        LXI  H,80F   ;     INTEGRAL
    1CBB   0F
    1CBC   08
110 1CBD   CD        CALL FOPD    ;       CONST. KI
    1CBE   13
    1CBF   1E
111 1CC0   CD        CALL MUPY    ;TS*KI
    1CC1   C1
    1CC2   10
112 1CC3   E5        PUSH H       ;FETCH
113 1CC4   21        LXI  H,815   ;       E(T)
    1CC5   15
    1CC6   08
114 1CC7   CD        CALL FOPR
    1CC8   0A
    1CC9   1E
115 1CCA   CD        CALL MUPY    ;TS*KI*E(T)=DF
    1CCB   C1
    1CCC   10
116 1CCD   D5        PUSH D       ;SAVE
117 1CCE   E5        PUSH H       ;     DF
118 1CCF   2A        LHLD 0818    ;FETCH
    1CD0   18
    1CD1   08
119 1CD2   EB        XCHG
120 1CD3   3A        LDA  081A    ;       E(T-1)
    1CD4   1A
    1CD5   08
121 1CD6   EE        XRI  80      ;       & SET
    1CD7   80
122 1CD8   6F        MOV  L,A     ;       -VE
123 1CD9   E5        PUSH H       ;FETCH
124 1CDA   21        LXI  H,815   ;       E(T)
    1CDB   15
    1CDC   08
125 1CDD   CD        CALL FOPR
    1CDE   0A
    1CDF   1E
126 1CE0   CD        CALL SUMM    ;E(T)-E(T-1)
    1CE1   03
    1CE2   10
127 1CE3   E5        PUSH H       ;FETCH
128 1CE4   21        LXI  H,80C   ;       KP
    1CE5   0C
    1CE6   08
129 1CE7   CD        CALL FOPR
    1CE8   0A
    1CE9   1E
130 1CEA   CD        CALL MUPY    ;(E(T)-E(T-1))*KP
    1CEB   C1
    1CEC   10
131 1CED   7D        MOV  A,L
132 1CEE   E1        POP  H       ;RESTORE
133 1CEF   C1        POP  B       ;         DF
134 1CF0   65        MOV  H,L
135 1CF1   6F        MOV  L,A
136 1CF2   CD        CALL SUMM    ;(E(T)-E(T-1))*KP+DF=P
    1CF3   03
    1CF4   10
137 1CF5   D5        PUSH D       ;SAVE
138 1CF6   E5        PUSH H       ;     P
139 1CF7   2A        LHLD 0818    ;FETCH E(T-1)
    1CF8   18
    1CF9   08
140 1CFA   EB        XCHG
141 1CFB   2A        LHLD 081A
    1CFC   1A
    1CFD   08
142 1CFE   65        MOV  H,L
143 1CFF   42        MOV  B,D
144 1D00   4B        MOV  C,E
145 1D01   CD        CALL SUMM    ;2E(T-1)
    1D02   03
    1D03   10
146 1D04   7D        MOV  A,L
147 1D05   EE        XRI  80
    1D06   80
148 1D07   6F        MOV  L,A     ;SET 2E(T-1)-VE
149 1D08   E5        PUSH H
150 1D09   21        LXI  H,815
    1D0A   15
    1D0B   08
```

```
151 1D0C   CD          CALL FOPR    ;FETCH E(T)
    1D0D   0A
    1D0E   1E
152 1D0F   CD          CALL SUMM    ;E(T)-2E(T-1)
    1D10   03
    1D11   10
153 1D12   E5          PUSH H
154 1D13   21          LXI  H,81B   ;FETCH
    1D14   1B
    1D15   03
155 1D16   CD          CALL FOPR    ;       E(T-2)
    1D17   0A
    1D18   1E
156 1D19   CD          CALL SUMM    ;E(T)-2E(T-1)+E(T-2)
    1D1A   03
    1D1B   10
157 1D1C   E5          PUSH H
158 1D1D   21          LXI  H,812   ;FETCH
    1D1E   12
    1D1F   08
159 1D20   CD          CALL FOPR    ;       KD
    1D21   0A
    1D22   1E
160 1D23   CD          CALL MUPY    ;(E(T)-2E(T-1)+E(T-2))*KD
    1D24   C1
    1D25   10
161 1D26   7D          MOV  A,L
162 1D27   2A          LHLD 0800    ;FETCH TS
    1D28   00
    1D29   08
163 1D2A   44          MOV  B,H
164 1D2B   4D          MOV  C,L
165 1D2C   6F          MOV  L,A
166 1D2D   26          MVI  H,0C2
    1D2E   C2
167 1D2F   CD          CALL COPR    ;SHIFT OPERATOR
    1D30   20
    1D31   1E
168 1D32   CD          CALL DVD     ;(E(T)-2E(T-1)+E(T-2))*KD/TS
    1D33   AE
    1D34   11
169 1D35   7D          MOV  A,L
170 1D36   E1          POP  H       ;RESTORE
171 1D37   C1          POP  B       ;       P
172 1D38   65          MOV  H,L
173 1D39   6F          MOV  L,A
174 1D3A   CD          CALL SUMM    ;((E(T)-2E(T-1)+E(T-2))*KD/TS)+P=DC
    1D3B   03
    1D3C   10
175 1D3D   3A          LDA  0805    ;FETCH O/P STATUS
    1D3E   05
    1D3F   08
176 1D40   E6          ANI  01
    1D41   01
177 1D42   C2          JNZ  BINO    ;IF BINARY
    1D43   5C
    1D44   1D
178 1D45   7D          MOV  A,L
179 1D46   2A          LHLD 0803    ;FETCH C
    1D47   03
    1D48   08
180 1D49   44          MOV  B,H
181 1D4A   4D          MOV  C,L
182 1D4B   6F          MOV  L,A
183 1D4C   26          MVI  H,0C0
    1D4D   C0
184 1D4E   CD          CALL SUMM    ;C+DC=C
    1D4F   03
    1D50   10
185 1D51   7D          MOV  A,L     ;O/P C
186 1D52   CD          CALL ALEX
    1D53   D1
    1D54   1D
187 1D55   21 EXAL:    LXI  H,0803
    1D56   03
    1D57   08
188 1D58   73          MOV  M,E     ;O/P L.S.D.
189 1D59   23          INX  H
190 1D5A   72          MOV  M,D     ;O/P M.S.D
191 1D5B   C9          RET
192 1D5C   7D BINO:    MOV  A,L     ;START B.C.D. TO BINARY CONV'N.
193 1D5D   CD          CALL ALEX    ;REARRANGE DC TO REQUIRED ORDER
    1D5E   D1
    1D5F   1D
194 1D60   7A          MOV  A,D
195 1D61   E6          ANI  0F0
    1D62   F0
196 1D63   0F          RRC          ;CONV. M.S.D.
197 1D64   47          MOV  B,A
198 1D65   0F          RRC
199 1D66   0F          RRC
```

```
200 1D67    80          ADD  B       ;M.S.D.*8+M.S.D.*2=M.S.D.*10
201 1D68    47          MOV  B,A
202 1D69    7A          MOV  A,D
203 1D6A    E6          ANI  0F
    1D6B    0F
204 1D6C    80          ADD  B
205 1D6D    07          RLC          ;CONV. M.S.D. & 2ND DIG.
206 1D6E    0E          MVI  C,0
    1D6F    00
207 1D70    57          MOV  D,A
208 1D71    87          ADD  A
209 1D72    DC          CC   CARY
    1D73    1C
    1D74    1E
210 1D75    F5          PUSH PSW     ;DOUBLE
211 1D76    79          MOV  A,C     ;  M.S.D.
212 1D77    87          ADD  A
213 1D78    4F          MOV  C,A
214 1D79    F1          POP  PSW
215 1D7A    87          ADD  A
216 1D7B    DC          CC   CARY
    1D7C    1C
    1D7D    1E
217 1D7E    82          ADD  D
218 1D7F    DC          CC   CARY
    1D80    1C
    1D81    1E
219 1D82    57          MOV  D,A
220 1D83    7B          MOV  A,E     ;CONV.  LAST DIGIT
221 1D84    0F          RRC
222 1D85    0F          RRC
223 1D86    0F          RRC
224 1D87    0F          RRC
225 1D88    E6          ANI  0F
    1D89    0F
226 1D8A    82          ADD  D
227 1D8B    DC          CC   CARY
    1D8C    1C
    1D8D    1E
228 1D8E    5F          MOV  E,A     ;CONV. N COMPLETED
229 1D8F    79          MOV  A,C     ;POSITION RESULT IN BC
230 1D90    07          RLC
231 1D91    07          RLC
232 1D92    07          RLC
233 1D93    07          RLC
234 1D94    47          MOV  B,A
235 1D95    7B          MOV  A,E
236 1D96    0F          RRC
237 1D97    0F          RRC
238 1D98    0F          RRC
239 1D99    0F          RRC
240 1D9A    5F          MOV  E,A
241 1D9B    E6          ANI  0F
    1D9C    0F
242 1D9D    B0          ORA  B
243 1D9E    47          MOV  B,A
244 1D9F    7B          MOV  A,E
245 1DA0    E6          ANI  0F0
    1DA1    F0
```



```
  1 1DA2    4F          MOV  C,A
  2 1DA3    7D          MOV  A,L     ;CHECK IF DC -VE
  3 1DA4    E6          ANI  80
    1DA5    80
  4 1DA6    2A          LHLD 803     ;FETCH C & REVERSE C
    1DA7    03
    1DA8    08
  5 1DA9    7C          MOV  A,H
  6 1DAA    65          MOV  H,L
  7 1DAB    6F          MOV  L,A
  8 1DAC    CA          JZ   MIDEL
    1DAD    BA
    1DAE    1D
  9 1DAF    09          DAD  B
 10 1DB0    DA          JC   EXMI
    1DB1    FF
    1DB2    1D
 11 1DB3    7C          MOV  A,H
 12 1DB4    65          MOV  H,L
 13 1DB5    6F          MOV  L,A
 14 1DB6    22          SHLD 803
    1DB7    03
    1DB8    08
 15 1DB9    C9          RET
 16 1DBA    7D MIDEL:   MOV  A,L     ;C-DC=C
 17 1DBB    91          SUB  C
 18 1DBC    6F          MOV  L,A
 19 1DBD    7C          MOV  A,H
 20 1DBE    98          SBB  B
 21 1DBF    67          MOV  H,A
 22 1DC0    7C          MOV  A,H
 23 1DC1    65          MOV  H,L
 24 1DC2    6F          MOV  L,A
```

```
25 1DC3  22          SHLD 803
   1DC4  03
   1DC5  08
26 1DC6  D0          RNC
27 1DC7  AF UFA:     XRA  A       ;CLEAR O/P
28 1DC8  32          STA  803
   1DC9  03
   1DCA  08
29 1DCB  3E          MVI  A,0A8   ;U/F ALARM
   1DCC  A8
30 1DCD  32          STA  8200
   1DCE  00
   1DCF  82
31 1DD0  C9          RET
32                   ;SUBROUTINE FOR SHIFTING A NUMBER ON DE & EXP. IN
33                   ;ACC. UNTIL EXP.=0
34 1DD1  47 ALEX:    MOV  B,A     ;SAVE ACC.
35 1DD2  E6          ANI  7F
   1DD3  7F
36 1DD4  C8          RZ
37 1DD5  FE          CPI  40
   1DD6  40
38 1DD7  C8          RZ
39 1DD8  DA          JC   AL1
   1DD9  EA
   1DDA  1D
40 1DDB  AF          XRA  A
41 1DDC  B2          ORA  D
42 1DDD  B3          ORA  E
43 1DDE  CA          JZ   AL1     ;IF DE=0
   1DDF  EA
   1DE0  1D
44 1DE1  78          MOV  A,B
45 1DE2  E6          ANI  80
   1DE3  80
46 1DE4  C2          JNZ  EXMI    ;IF DC O/F
   1DE5  FF
   1DE6  1D
47 1DE7  C3          JMP  UFA     ;IF DC U/F
   1DE8  C7
   1DE9  1D
48 1DEA  CD AL1:     CALL RDER
   1DEB  ED
   1DEC  12
49 1DED  CD          CALL RDER
   1DEE  ED
   1DEF  12
50 1DF0  CD          CALL RDER
   1DF1  ED
   1DF2  12
51 1DF3  CD          CALL RDER
   1DF4  ED
   1DF5  12
52 1DF6  7A          MOV  A,D
53 1DF7  E6          ANI  0F
   1DF8  0F
54 1DF9  57          MOV  D,A
55 1DFA  78          MOV  A,B     ;RESTORE ACC.
56 1DFB  3D          DCR  A
57 1DFC  C3          JMP  ALEX
   1DFD  D1
   1DFE  1D
58 1DFF  3E EXMI:    MVI  A,0A7   ;O/F ALARM
   1E00  A7
59 1E01  32          STA  8200
   1E02  00
   1E03  82
60 1E04  3E          MVI  A,0FF
   1E05  FF
61 1E06  32          STA  0803
   1E07  03
   1E08  08
62 1E09  C9          RET
63                   ;SUBROUTINE FOR POSITIONING OPERATOR (BC&H)
64 1E0A  4E FOPR:    MOV  C,M     ;FETCH L.S.D.
65 1E0B  23          INX  H
66 1E0C  46          MOV  B,M     ;FETCH M.S.D.
67 1E0D  23          INX  H
68 1E0E  7E          MOV  A,M     ;FETCH EXP.
69 1E0F  E1          POP  H
70 1E10  E3          XTHL         ;RESTORE PREVIOUS HL&PC
71 1E11  67          MOV  H,A
72 1E12  C9          RET
73                   ;SUBROUTINE FOR POSITIONING OPERAND (DE&L)
74 1E13  5E FOPD:    MOV  E,M     ;FETCH L.S.D.
75 1E14  23          INX  H
76 1E15  56          MOV  D,M     ;FETCH M.S.D.
77 1E16  23          INX  H
78 1E17  7E          MOV  A,M     ;FETCH EXP.
79 1E18  E1          POP  H       ;RESTORE PREVIOUS
80 1E19  E3          XTHL         ;   HL & PC
81 1E1A  6F          MOV  L,A
82 1E1B  C9          RET
```

```
 83                     ;SUBROUTINE FOR CARRY OVER
 84 1E1C     F5 CARY:   PUSH PSW
 85 1E1D     0C         INR  C
 86 1E1E     F1         POP  PSW
 87 1E1F     C9         RET
 88                     ;SUBROUTINE FOR SHIFTING OPERATOR LEFT UNTILL
 89                     ;M.S.D. IS NON-ZERO.
 90 1E20     78 COPR:   MOV      A,B
 91 1E21     E6         ANI      0F0
    1E22     F0
 92 1E23     C0         RNZ                 ;IF M.S.D. IS NON-ZERO
 93 1E24     CD         CALL     SBCL
    1E25     44
    1E26     1E
 94 1E27     CD         CALL     SBCL
    1E28     44
    1E29     1E
 95 1E2A     CD         CALL     SBCL
    1E2B     44
    1E2C     1E
 96 1E2D     CD         CALL     SBCL
    1E2E     44
    1E2F     1E
 97 1E30     7C         MOV      A,H
 98 1E31     E6         ANI      7F
    1E32     7F
 99 1E33     FE         CPI      41
    1E34     41
100 1E35     DA         JC       NVE       ;IF EXP.=0 OR -VE
    1E36     3C
    1E37     1E
101 1E38     25         DCR      H
102 1E39     C3         JMP      COPR
    1E3A     20
    1E3B     1E
103 1E3C     24 NVE:    INR      H
104 1E3D     7C         MOV      A,H
105 1E3E     E6         ANI      0BF       ;SET EXP.=-VE
    1E3F     BF
106 1E40     67         MOV      H,A
107 1E41     C3         JMP      COPR
    1E42     20
    1E43     1E
108                     ;SUBROUTINE FOR SHIFTING BC LEFT.
109 1E44     AF SBCL:   XRA      A         ;CLEAR CY FLAG
110 1E45     79         MOV      A,C
111 1E46     17         RAL
112 1E47     4F         MOV      C,A
113 1E48     78         MOV      A,B
114 1E49     17         RAL
115 1E4A     47         MOV      B,A
116 1E4B     C9         RET
117                     .END
AL1     1DEA      ALEX    1DD1      BCDI    1C78      BINO    1D5C

CARY    1E1C      COPR    1E20      DVD     11AE      EXAL    1D55

EXMI    1DFF      EXPF    1C24      FOPD    1E13      FOPR    1E0A

MIDEL   1DBA      MUPY    10C1      MVA     1C83      NVE     1E3C

RDER    12ED      SBCL    1E44      STHA    1C1D      SUMM    1003

SUSB    1C0A      UFA     1DC7
```

APPENDIX 11

Listing of the Non-linear Adaptive Controller Program.

```
                .TITLE 'MODIFIED 2-TERM CONTROLLER'
                .HEX
                BSU=135E
                BHU=1365
                ETL=827
                ETH=828
                ETML=82A
                ETMH=82B
                YKL=82D
                YKH=82E
                CL=830
                CH=831
                KIL=833
                KIH=834
                KPL=836
                KPH=837
1800    00      .ORG 1800
1800    AF              XRA     A
1801    21              LXI     H,ETL
1802    27
1803    08
1804    77              MOV     M,A     ;CLEAR E(T)
1805    23              INX     H
1806    77              MOV     M,A
1807    23              INX     H
1808    23              INX     H
1809    77              MOV     M,A     ;CLEAR E(T-1)
180A    23              INX     H
180B    77              MOV     M,A
180C    23              INX     H
180D    23              INX     H
180E    36              MVI     M,20    ;SET Y(K) EST.
180F    20
1810    23              INX     H
1811    77              MOV     M,A
1812    23              INX     H
1813    23              INX     H
1814    77              MOV     M,A     ;CLEAR O/P (C)
1815    23              INX     H
1816    77              MOV     M,A
1817    23              INX     H
1818    23              INX     H
1819    36              MVI     M,48    ;SET KI=BI
181A    48
181B    23              INX     H
181C    77              MOV     M,A
181D    23              INX     H
181E    23              INX     H
181F    36              MVI     M,0C0   ;SET DP=BP
1820    C0
1821    23              INX     H
1822    36              MVI     M,02
1823    02
1824    3E              MVI     A,37    ;PROGRAM COUNTER
1825    37
1826    32              STA     8007
1827    07
1828    80
1829    3E              MVI     A,50    ;SET TIMER TO 1/2  SEC.
182A    50
182B    32              STA     8004
182C    04
182D    80
182E    32              STA     800
182F    00
1830    08
1831    3E              MVI     A,0
1832    00
1833    32              STA     8004
1834    04
1835    80
1836    32              STA     801
1837    01
1838    08
1839    C3              JMP     INTR
183A    00
183B    14
1400    00      .ORG 1400
1400    3A      INTR:   LDA     806     ;READ LEVEL
1401    06
1402    08
1403    2F              CMA             ;CONVERT TO BINARY
1404    07              RLC             ;SCALE UP I/P
1405    07              RLC
1406    07              RLC
1407    5F              MOV     E,A     ;LOW PASS FILTER
1408    3A              LDA     YKL     ;K=.5 I.E. KY=Y/2
1409    2D
140A    08
140B    0F              RRC
```

```
66 1400  E6          ANI     7F
   1401  7F
67 1402  83          ADD     E
68 1403  32          STA     YKL
   1404  2D
   1405  08
69 1406  3A          LDA     ETL     ;STIFT ERROR E(T-1)
   1407  27
   1408  08
70 1409  32          STA     ETML
   140A  2A
   140B  08
71 140C  3A          LDA     YKL     ;CALCULATE E(T)
   140D  2D
   140E  08
72 140F  D6          SUI     20
   1410  20
73 1411  F2          JP      PVE     ;CONVERT TO SIGNED BINARY
   1412  23
   1413  14
74 1414  2F          CMA             ;SIGN -VE
75 1415  C6          ADI     81      ;TWO'S COMPL.
   1416  81
76 1417  32 PVE:     STA     ETL     ;STORE E(T)
   1418  27
   1419  08
77 141A  21          LXI     H,ETML
   141B  2A
   141C  08
78 141D  86          ADD     M       ;/E(T)/+/E(T-1)/
79 141E  E6          ANI     7F
   141F  7F
80 1420  FE          CPI     31
   1421  31
81 1422  DA          JC      WIR     ;IF NOT> 3*W (W=SCALING)
   1423  5F
   1424  14
82 1425  3A          LDA     ETL
   1426  27
   1427  08
83 1428  E6          ANI     80
   1429  80
84 142A  C2          JNZ     NVE     ;IF -VE
   142B  46
   142C  14
85 142D  3E          MVI     A,0FC
   142E  FC
86 142F  32          STA     CH
   1430  31
   1431  08
87 1432  3E          MVI     A,0
   1433  00
88 1434  32          STA     CL
   1435  30
   1436  08
89 1437  C3          JMP     SCT
   1438  4D
   1439  14
90 143A  AF NVE:     XRA     A
91 143B  32          STA     CL
   143C  30
   143D  08
92 143E  32          STA     CH
   143F  31
   1440  08
93 1441  21 SCT:     LXI     H,KIL
   1442  33
   1443  08
94 1444  36          MVI     M,48    ;KI=BI
   1445  48
95 1446  23          INX     H
96 1447  36          MVI     M,0
   1448  00
97 1449  23          INX     H
98 144A  23          INX     H
99 144B  36          MVI     M,0C0   ;KP=BP
   144C  C0
100 144D 23          INX     H
101 144E 36          MVI     M,02
   144F  02
102 1450 C3          JMP     TTC
   1451  A0
   1452  14
103 1453 E6 WIR:     ANI     7F
   1454  7F
104 1455 FE          CPI     08
   1456  08
105 1457 D2          JNC     TTC     ;IF /E(T)/ NOT< .5*W
   1458  A0
   1459  14
106 145A 3A          LDA     KIL
   145B  33
   145C  08
```

```
107 1469    FE          CPI     28
    146A    28
108 146B    D2          JNC     CKI     ;IF KI>=AI
    146C    73
    146D    14
109 146E    3E          MVI     A,28    ;KI=AI
    146F    28
110 1470    C3          JMP     CKI1
    1471    77
    1472    14
111 1473    CA CKI:     JZ      CKI1
    1474    77
    1475    14
112 1476    3D          DCR     A       ;KI - 1
113 1477    32 CKI1:    STA     KIL
    1478    33
    1479    08
114 147A    3A          LDA     KPH     ;FETCH M.S.D. OF KP
    147B    37
    147C    08
115 147D    FE          CPI     0
    147E    00
116 147F    C2          JNZ     CKP
    1480    95
    1481    14
117 1482    3A          LDA     KPL     ;FETCH L.S.D. OF KP
    1483    36
    1484    08
118 1485    FE          CPI     0C0
    1486    C0
119 1487    D2          JNC     CKP1
    1488    92
    1489    14
120 148A    3E          MVI     A,0C0
    148B    C0
121 148C    32          STA     KPL     ;KP=AP
    148D    36
    148E    08
122 148F    C3          JMP     TTC
    1490    A0
    1491    14
123 1492    CA CKP1:    JZ      TTC
    1493    A0
    1494    14
124 1495    21 CKP:     LXI     H,KPL   ;DECREMENT KP
    1496    36
    1497    08
125 1498    5E          MOV     E,M
126 1499    23          INX     H
127 149A    56          MOV     D,M
128 149B    1B          DCX     D
129 149C    1B          DCX     D
130 149D    72          MOV     M,D
131 149E    2B          DCX     H
132 149F    73          MOV     M,E
133 14A0    3A TTC:     LDA     KIL     ;FETCH L.S.D. OF KI
    14A1    33
    14A2    08
134 14A3    0F          RRC
135 14A4    0F          RRC
136 14A5    0F          RRC
137 14A6    0F          RRC
138 14A7    E6          ANI     0F      ;KI/16
    14A8    0F
139 14A9    6F          MOV     L,A
140 14AA    26          MVI     H,0
    14AB    00
141 14AC    3A          LDA     ETL     ;FETCH E(T)
    14AD    27
    14AE    08
142 14AF    E6          ANI     7F
    14B0    7F
143 14B1    5F          MOV     E,A
144 14B2    16          MVI     D,0
    14B3    00
145 14B4    CD          CALL    BMU     ;KIE=E(T)*KI
    14B5    65
    14B6    13
146 14B7    44          MOV     B,H
147 14B8    4D          MOV     C,L
148 14B9    21          LXI     H,ETL
    14BA    27
    14BB    08
149 14BC    3A          LDA     ETML
    14BD    2A
    14BE    08
150 14BF    AE          XRA     M
151 14C0    F2          JP      EME     ;IF SIGN EQ.
    14C1    D6
    14C2    14
152 14C3    3A          LDA     ETML
    14C4    2A
    14C5    08
```

```
153 14C6    E6          ANI     7F
    14C7    7F
154 14C8    83          ADD     E       ;E(T)+E(T-1)
155 14C9    5F          MOV     E,A     ;SAVE RESULT
156 14CA    3A          LDA     ETL
    14CB    27
    14CC    08
157 14CD    E6          ANI     80
    14CE    80
158 14CF    7B          MOV     A,E     ;RETORE RESULT
159 14D0    C2          JNZ     RN      ;IF E(T) -VE
    14D1    F6
    14D2    14
160 14D3    C3          JMP     KPN
    14D4    F7
    14D5    14
161 14D6    3A EME:     LDA     ETML
    14D7    2A
    14D8    08
162 14D9    E6          ANI     80
    14DA    80
163 14DB    CA          JZ      TMO     ;E(T-1)=+VE
    14DC    EC
    14DD    14
164 14DE    3A          LDA     ETML
    14DF    2A
    14E0    08
165 14E1    E6          ANI     7F
    14E2    7F
166 14E3    93          SUB     E       ;E(T-1)-E(T)
167 14E4    F2          JP      KPN     ;IF RESULT +VE
    14E5    F7
    14E6    14
168 14E7    2F          CMA
169 14E8    3C          INR     A
170 14E9    C3          JMP     RN      ;IF RESULT -VE
    14EA    F6
    14EB    14
171 14EC    7B TMO:     MOV     A,E
172 14ED    21          LXI     H,ETML
    14EE    2A
    14EF    08
173 14F0    96          SUB     M       ;E(T)-E(T-1)
174 14F1    F2          JP      KPN     ;IF RESULT +VE
    14F2    F7
    14F3    14
175 14F4    2F          CMA
176 14F5    3C          INR     A
177 14F6    37 RN:      STC
178 14F7    F5 KPN:     PUSH    PSW
179 14F8    21          LXI     H,KPL
    14F9    36
    14FA    08
180 14FB    5E          MOV     E,M
181 14FC    23          INX     H
182 14FD    56          MOV     D,M
183 14FE    6F          MOV     L,A
184 14FF    26          MVI     H,0
    1500    00
185 1501    CD          CALL    BMU     ;KPE=KP*(E(T)-E(T-1))
    1502    65
    1503    13
186 1504    50          MOV     D,B
187 1505    59          MOV     E,C
188 1506    F1          POP     PSW
189 1507    DA          JC      KPEN    ;IF KPE -VE
    1508    18
    1509    15
190 150A    3A          LDA     ETL
    150B    27
    150C    08
191 150D    E6          ANI     80
    150E    80
192 150F    CA          JZ      PEI     ;IF E(T) +VE
    1510    31
    1511    15
193 1512    CD          CALL    BSU     ;KPE-KIE
    1513    5E
    1514    13
194 1515    C3          JMP     PC
    1516    24
    1517    15
195 1518    3A KPEN:    LDA     ETL
    1519    27
    151A    08
196 151B    E6          ANI     80
    151C    80
197 151D    C2          JNZ     PEI     ;IF E(T) -VE
    151E    31
    151F    15
198 1520    EB          XCHG
199 1521    CD          CALL    BSU     ;KIE-KPE
    1522    5E
    1523    13
```

```
200 1524    D2 PC:    JNC     PDC     ;IF +VE
    1525    4B
    1526    15
201 1527    7C        MOV     A,H     ;2'S COMPL.
202 1528    2F        CMA
203 1529    67        MOV     H,A
204 152A    7D        MOV     A,L
205 152B    2F        CMA
206 152C    6F        MOV     L,A
207 152D    23        INX     H
208 152E    C3        JMP     NDC
    152F    35
    1530    15
209 1531    19 PEI:   DAD     D
210 1532    CA        JZ      PDC     ;IF DC +VE
    1533    4B
    1534    15
211 1535    3A NDC:   LDA     CL
    1536    30
    1537    08
212 1538    5F        MOV     E,A
213 1539    3A        LDA     CH
    153A    31
    153B    08
214 153C    57        MOV     D,A
215 153D    EB        XCHG
216 153E    CD        CALL    BSU     ;C=C-DC
    153F    5E
    1540    13
217 1541    D2        JNC     SC
    1542    5B
    1543    15
218 1544    26        MVI     H,0     ;C=0 IF C=-VE
    1545    00
219 1546    2E        MVI     L,0
    1547    00
220 1548    C3        JMP     SC
    1549    5B
    154A    15
221 154B    3A PDC:   LDA     CL
    154C    30
    154D    08
222 154E    5F        MOV     E,A
223 154F    3A        LDA     CH
    1550    31
    1551    08
224 1552    57        MOV     D,A
225 1553    19        DAD     D       ;C=C+DC
226 1554    D2        JNC     SC
    1555    5B
    1556    15
227 1557    26        MVI     H,0FC   ;C=MAX. IF C=OVF.
    1558    FC
228 1559    2E        MVI     L,0
    155A    00
229 155B    EB SC:    XCHG
230 155C    21        LXI     H,CL    ;STORE C
    155D    30
    155E    08
231 155F    73        MOV     M,E
232 1560    23        INX     H
233 1561    72        MOV     M,D
234 1562    7A        MOV     A,D     ;O/P  C/2**8
235 1563    32        STA     803
    1564    03
    1565    08
236 1566    C9        RET             ;WAIT FOR INTERRUPT
237             .END
```

BMU	1365	BSU	135E	CH	0831	CKI	1473
CKI1	1477	CKP	1495	CKP1	1492	CL	0830
EME	14D6	ETH	0828	ETL	0827	ETMH	082B
ETML	082A	INTR	1400	KIH	0834	KIL	0833
KPEN	1518	KPH	0837	KPL	0836	KPN	14F7
NDC	1535	NVE	1446	PC	1524	PDC	154B
PEI	1531	PVE	1423	RN	14F6	SC	155B
SCT	144D	TMO	14EC	TTC	14A0	WIR	145F
YKH	082E	YKL	082D				

*